Hardness and Independence of Polynomials

Prerona Chatterjee

Hardness and Independence of Polynomials

A thesis submitted to the

Tata Institute of Fundamental Research, Mumbai

for the degree of Doctor of Philosophy

by

Prerona Chatterjee

School of Technology and Computer Science

Tata Institute of Fundamental Research, Mumbai

December, 2021

Final Version Submitted in March, 2022

DECLARATION

This thesis, titled **“Hardness and Independence of Polynomials**" is a presentation of my original research work. Wherever contributions of others are involved, every effort is made to indicate this clearly, with due reference to the literature, and acknowledgement of collaborative research and discussions.

The work was done under the guidance of **Dr. Ramprasad Saptharishi** at the Tata Institute of Fundamental Research, Mumbai.

Prerona Chatterjee

Prerona Chatterjee

In my capacity as supervisor of the candidate’s thesis, I certify that the above statements are true to the best of my knowledge.

Ramprasad

Ramprasad Saptharishi

Date: 2022-03-17

Contents

List of Publications

1. Constructing Faithful Homomorphisms over Fields of Finite Characteristic. Joint work with Ramprasad Saptharishi. Preliminary version in the proceedings of FSTTCS 2019 [CS19]. Full version currently under review.
2. A Quadratic Lower Bound for Algebraic Branching Programs and Formulas. Joint work with Mrinal Kumar, Adrian She and Ben Lee Volk. Preliminary version in the proceedings of CCC 2020 [Cha+20]. Full version to appear in Computational Complexity.
3. Separating ABPs and Some Structured Formulas in the Non-Commutative Setting. In the proceedings of CCC 2021 [Cha21].

List of Figures

1 Introduction

Theory of Computation is the study of formalising the notion of computation. In the early years of the field, the focus was on trying to understand what problems can or cannot be solved computationally. However, it quickly became clear that this division is too coarse. It may be the case that a certain problem can be solved algorithmically but by the time the algorithm outputs the answer, the answer is no longer useful. This led to the birth of Computational Complexity Theory.

The aim of complexity theory is to understand *efficient computation* in terms of the resources (like time or space) required by natural computational models to solve the given problem. The work in this thesis belongs to a subfield of complexity theory that deals with problems which are algebraic in nature. Problems in Algebraic Complexity Theory are typically of the form "How hard is it to perform a given computational task Y on a given algebraic object X?" In this thesis, we are interested in some specific questions of this type.

The thesis is divided into two parts. In the first part, we ask how hard it is to compute a given polynomial for a given computational model, whereas in the second part we consider the following two questions and see how they are related. The first is how hard it is to check whether a given set of polynomials are *algebraically independent* or not, and the second is how hard it is to check whether a given computational model computes the zero polynomial or not.

Before stating the questions more formally, let us define the models of computation that we will be studying. An *algebraic circuit* is a very natural (and the most general) algebraic computational model. Informally, it is a computational device which is given a set of indeterminates $\{x_1, \ldots, x_n\}$, and it can use additions and multiplications (as well as field scalars) to compute a polynomial $f \in \mathbb{F}[x_1, \ldots, x_n]$ (see Figure 1.1). The underlying structure is that of a rooted directed acyclic graph and the size of a circuit is defined to be the number of operations the circuit performs (or the number of nodes in the underlying graph). If the underlying graph is only allowed to be a tree, then the model of computation is that of an *algebraic formula*.

Another important model of computation is that of an *algebraic branching program*, or ABP (see Figure 1.2). The computational power of ABPs lie in between that of for-

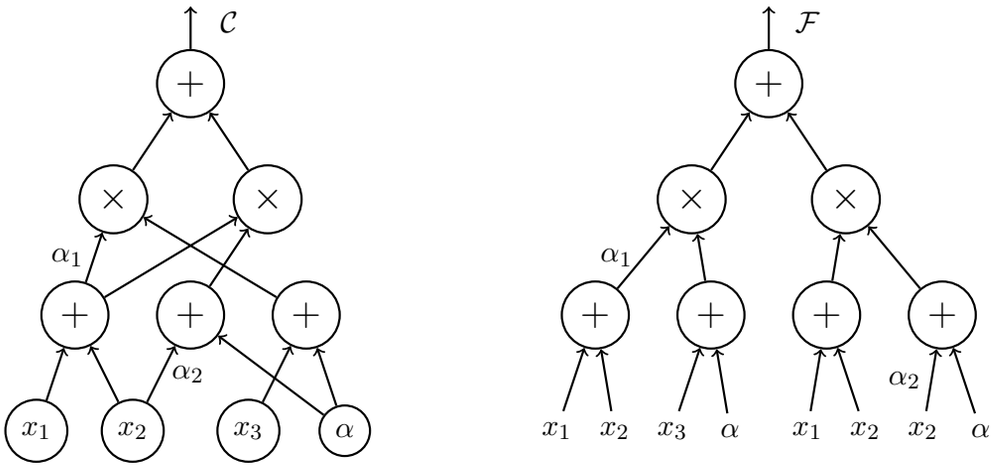


Figure 1.1.: Algebraic circuits and formulas. The polynomial being computed in both cases is $\alpha_1(x_1 + x_2)(x_3 + \alpha) + (x_1 + x_2)(\alpha_2 x_2 + \alpha)$ where α, α_1, α_2 are field constants.

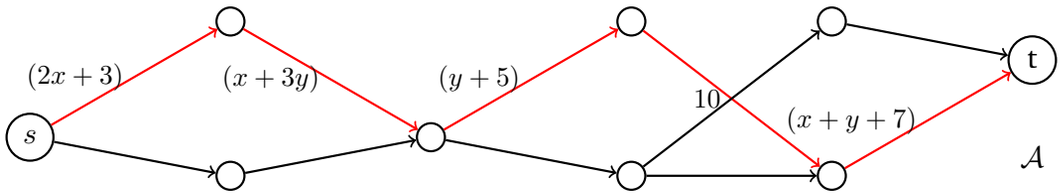


Figure 1.2.: Algebraic Branching Programs (ABPs). Layered graph with designated start (s) and target (t) vertices in which every edge is labelled by an affine linear form in the underlying variables. The polynomial being computed by an s – t path is the product of its edge labels (for example, the highlighted path computes $10(2x + 1)(x + 3y)(y + 5)(x + y + 7)$). The polynomial computed by the ABP is the sum of the polynomials computed by the various s – t paths.

mulas and circuits, and therefore can be used as an intermediate step for extending results that are known for formulas to circuits. Its importance is further enhanced because several important polynomials, including the determinant polynomial, is *complete* for the class of polynomials that can be computed efficiently by ABPs.

Lower Bounds in Algebraic Circuit Complexity The sub-area of algebraic circuit complexity forms an integral part of algebraic complexity theory. In it, the central question asks whether there is an *explicit* n-variate, degree d polynomial that can not be computed by algebraic circuits of size $\mathrm{poly}(n, d)$. In other words, can we give a super-polynomial lower bound on the size of any algebraic circuit computing some explicit polynomial? This is the algebraic analogue of the famed P vs NP problem.

In the first part of the thesis, we are interested in questions of a similar flavour.

Concretely, we address the following questions.

1. Does there exist an explicit n-variate, degree d polynomial that cannot be computed by algebraic branching programs of size $\mathrm{poly}(n, d)$?

2. Does there exist an explicit n-variate, degree d polynomial that cannot be computed by algebraic formulas of size $\mathrm{poly}(n, d)$?

3. In the *multilinear* setting, strong lower bounds are known against formulas but only weak bounds are known against circuits. Is there a reason for this?

4. Does there exist an n-variate, degree d polynomial that can be computed by algebraic branching programs of size $\mathrm{poly}(n, d)$ but cannot be computed by algebraic formulas of size $\mathrm{poly}(n, d)$ in the *non-commutative* setting?

With respect to the first two questions, we show a quadratic lower bound against both algebraic branching programs and algebraic formulas. On the other hand, with respect to the second question, we show that lower bounds in the setting mentioned that are just slightly stronger than the ones known have surprising implications. Finally, with respect to the last question, we show a tight separation between the powers of ABPs and some structured formulas in the setting mentioned.

Formal statements of these questions and our results are described in section 1.2.

Algebraic Independence and Faithful Homomorphisms A given set of polynomials $\{f_1, \ldots, f_m\}$ is said to be algebraically dependent if there is some non-zero polynomial combination of them that is zero. For example, if $f_1 = x$, $f_2 = y$ and $f_3 = x^2 + y^2$, then $A = z_1^2 + z_2^2 - z_3$ is an *annihilator*.

A natural question at this point is whether there is an efficient algorithm to check whether a given set of polynomials are algebraically independent or not. Apart from this being an interesting question in its own right, the concept of algebraic independence has a connection with another important problem in algebraic complexity theory, namely the problem of Polynomial Identity Testing or PIT.

Given an algebraic circuit as input, the PIT problem asks whether the polynomial computed by it is identically zero. It is well known that this problem has an efficient randomised polynomial-time algorithm. A central algorithmic question in Algebraic Complexity Theory is whether this can be derandomised.

In section 1.3 we discuss the results known with respect to efficiently testing algebraic independence of a given set of polynomials and study its connections to the derandomising the problem of PIT via the concept of *faithful homomorphisms*.

Before moving to the details, let us go over some preliminary notions that we will require throughout the thesis.

1.1 Notations and Preliminaries

We denote by $\mathbf{x}$ the vector of variables $(x_1, \ldots, x_n)$, where n is understood from the context. Similarly we use $\mathbf{0}$ to denote the n-dimensional vector $(0, 0, \ldots, 0)$.

Algebraic Models of Computation

Let us recall the definitions of the models of computation we are interested in.

Algebraic Circuits and Formulas. An algebraic circuit is a rooted directed acyclic graph in which the leaves are labelled by indeterminates or field constants and the internal nodes are labelled by addition ($+$) or multiplication ($\times$). Thus every node in the circuit naturally computes a polynomial and the polynomial computed at the root is said to be the polynomial computed by the circuit. The size of a circuit is the number of operations the circuit performs. If the underlying graph is only allowed to be a tree, then the model of computation is that of an algebraic formula.

Algebraic Branching Programs. An algebraic branching program is a layered graph in which the first and the last layer only contains a single vertex each called s and t respectively. Edges in an ABP are present between consecutive layers and each edge is labelled by an affine linear polynomial over the underlying variables. The polynomial computed by any s–t path is the product of the edge labels on that path and the polynomial computed by the ABP is the sum of the polynomials computed by the various s–t paths. The size of the ABP is the number of vertices in it.

Multilinear Models of Computation Multilinear polynomials are those in which the individual degree of every variable in any monomial is at most one. Many of the well-studied polynomials have this property, and therefore one of the natural restrictions that is studied in the field is that of multilinearity.

A *multilinear circuit* is one in which the polynomial computed at each of its gate is multilinear. A *multilinear formula* can be defined in a similar way. Another syntactic way of defining multilinear models is the following. A *syntactically multilinear circuit* is one in which for any multiplication gate $v = v_1 \times v_2$, there is no variable that has a path to both v_1 and v_2. Similarly *syntactically multilinear formulas* are those which

have the above property. Raz [Raz09] showed that any multilinear formula is also syntactically multilinear, but this is not clear in the case for circuits.

Multilinear algebraic branching programs are those in which the polynomial computed between any two of its vertices is multilinear. Whereas *syntactically multilinear ABPs* are those in which no variable occurs more than once in any $s - t$ path.

Non-Commutative Models of Computation Given an algebraic circuit, if the multiplication gate is additionally given an order in which its inputs are multiplied, then the model of computation is that of a non-commutative circuit. That is, in this setting, the circuit cannot assume that $xy = xy$ for indeterminates x and y. Therefore polynomials computed by such circuits belong to the non-commutative polynomial ring denoted by $\mathbb{F}\langle x_1, \ldots, x_n \rangle$. Non-commutative formulas and ABPs can be defined analogously.

Constant Depth Circuits The depth of any circuit is defined to be the length of the longest path between any leaf and its root. Constant depth circuits, as the name suggests, are those circuits whose depth is constant.

Important Algebraic Classes

Similar to any other branch in complexity theory, in algebraic circuit complexity we define various classes of polynomials depending on how efficiently they can be computed. We mention the classes that are relevant to us.

The set of n-variate, degree $\mathrm{poly}(n)$ polynomials that can be computed by circuits of size $\mathrm{poly}(n)$ form the class VP — the algebraic analogue of the class P. Similarly, the class VBP consists of n-variate, degree d polynomials that can be computed by ABPs of size $\mathrm{poly}(n, d)$. Finally, the class VF is the set of n-variate, degree d polynomials that can be computed by formulas of size $\mathrm{poly}(n, d)$.

Analogous to the class NP, the class VNP is defined to be the set of all polynomial $f(x_1, \ldots, x_n)$ such that there exists a polynomial $g(x_1, \ldots, x_n, y_1, \ldots, y_m) \in \mathsf{VP}$ with $m = \mathrm{poly}(n)$ such that

$$f(x_1, \ldots, x_n) = \sum_{(y_1, \ldots, y_m) \in \{0,1\}^m} g(x_1, \ldots, x_n, y_1, \ldots, y_m).$$

In particular, Valiant [Val79] showed that for a polynomial f, given a monomial, if we can compute its coefficient in f polynomial time, then f is in VNP.

Both VP (the class of efficiently computable polynomials) and VNP (the class of efficiently definable polynomials) were formally defined by Valiant [Val79].

Non-commutative analogues of these classes, denoted by $\mathsf{VP}_{\mathrm{nc}}$, $\mathsf{VBP}_{\mathrm{nc}}$, $\mathsf{VF}_{\mathrm{nc}}$ and $\mathsf{VNP}_{\mathrm{nc}}$ respectively, are defined in a similar way.

1.2 Lower Bounds in Algebraic Circuit Complexity

The central question in Algebraic Circuit Complexity is that of proving lower bounds for explicit polynomials against various algebraic models of computation. In the most general form, this is the well-known VP vs. VNP question, which asks whether every *explicit* polynomial has a polynomial-size algebraic circuit.

Before we start looking for explicit lower bounds, let us ask ourselves whether hard polynomials exist. The answer to this question is *yes*. In fact most polynomials are hard. Over finite fields, it can be shown using usual counting techniques that the number of n-variate, degree d polynomials that have $\mathrm{poly}(n, d)$-sized circuits is much smaller than the total number of such polynomials. On the other hand, over infinite fields, such a statement is shown by counting dimensions.

In fact, it can also be shown that over every field, for any n, d, there exist n-variate, degree d polynomials with zero-one coefficients that cannot be computed by circuits of size $\mathrm{poly}(n, d)$ [SY10, Theorem 3.1]. Therefore, as mentioned earlier, the ultimate goal in algebraic circuit complexity is to find such a hard polynomial explicitly. Even though we have made great progress in restricted settings, in the general setting we seem to be quite far from achieving this goal.

General Models of Computation

It is trivial to give an explicit n-variate polynomial which requires circuits of size $\Omega(n)$. It is also not hard to show that a degree-d polynomial requires fan-in two circuits of size $\Omega(\log d)$, since the degree can at most double in each operation. Thus a lower bound of $\max\{n, \log d\} = \Omega(n + \log d)$ for an n-variate degree-d polynomial can be obtained easily. A major result of Baur and Strassen [Str73a; BS83] gives an explicit n-variate degree-d polynomial which requires circuits of size at least $\Omega(n \cdot \log d)$.

On the one hand, this is impressive since when $d = \mathrm{poly}(n)$, this gives a lower bound which is super-linear in n. Such lower bounds for explicit functions in the

analogous model of *boolean* circuits are a long-standing and important open problem in boolean circuit complexity. On the other hand, this lower bound is barely super-linear, when ideally we hope to prove super-polynomial or even exponential lower bounds. Despite decades of work, however, this lower bound has not been improved, even though it has been reproved (using different techniques [Smo97; Ben83]).

In the case of formulas, we know slightly better lower bounds. Kalorkoti [Kal85] has shown how to adapt Nechiporuk's method [Nec66], originally developed for boolean formulas, to prove an $\Omega(n^2/\log n)$ lower bound for an n-variate multilinear polynomial. We study the model of formulas in greater detail in chapter 3, where we show an $\Omega(n^2)$ lower bound. In particular, we show the following statement.

Theorem 1.2.1. *Let $n \in \mathbb{N}$ and let $\mathbb{F}$ be a field of characteristic greater than $0.1n$. Then any algebraic formula over $\mathbb{F}$ computing the elementary symmetric polynomial*

$$\mathrm{ESYM}_{n,0.1n}(\mathbf{x}) = \sum_{S\subseteq[n],|S|=0.1n} \prod_{j\in S} x_j \,,$$

is of size $\Omega(n^2)$.

Even though the improvement is only logarithmic, it is interesting to note that the techniques of Nechiporuk and Kalorkoti cannot be used to give a lower bound better than $\Omega(n^2/\log n)$.

Another point to note is that Baur and Strassen [BS83] proved an *upper bound* of $O(n\log n)$[1] on the circuit complexity of the elementary symmetric polynomials. So our lower bound implies a super-linear separation between the formula complexity and the circuit complexity of an explicit multilinear polynomial family.

For the intermediate model of algebraic branching programs, however, the result of Baur and Strassen continued to be the best lower bound known till before our work. A detailed discussion can be found in chapter 2, where we show a quadratic lower bound against algebraic branching programs for a very simple polynomial. The formal statement is as follows.

Theorem 1.2.2. *Let $\mathbb{F}$ be a field and $n \in \mathbb{N}$ such that $\mathrm{char}(\mathbb{F}) \nmid n$. Then any algebraic branching program over $\mathbb{F}$ computing the polynomial $\sum_{i=1}^{n} x_i^n$ is of size at least $\Omega(n^2)$.*

[1] The key step in the proof refers to a paper that is written in German and we were unable to reconstruct it. Ramprasad, however, told us about an $O(n\log^2 n)$ upper bound which can also be found on cs.stackexchange.

With a lack of techniques known to work against general models of computation, most of the works in algebraic circuit complexity deals with restricted models of computation. In this regard, great progress has been made by the community. For many natural restricted models, exponential or at least super-polynomial lower bounds are known.

We refer the reader to some excellent surveys [SY10; Sap15] for a comprehensive overview of lower bounds in algebraic complexity. Here we mention some of the widely studied restrictions that are relevant to our work.

Multilinear Models of Computation

In the multilinear setting, we know very good lower bounds against formulas. Raz [Raz09] showed that any multilinear formula computing the permanent or determinant of an $n \times n$ matrix must have size at least $n^{\Omega(\log n)}$.

Subsequently, Raz [Raz06] showed an explicit polynomial that can be computed by a polynomial size syntactically multilinear circuit but any multilinear formula computing it must have size at least $n^{\Omega(\log n)}$. Later Dvir, Malod, Perifel and Yehudayoff [Dvi+12] showed that in fact there exists an explicit polynomial that can be computed by polynomial size multilinear algebraic branching programs, but cannot be computed by multilinear formulas of size $n^{o(\log n)}$. A point to note, however, is that it is not clear whether the polynomial defined by Dvir *et al.* can be computed by a syntactically multilinear algebraic branching program.

Our understanding of multilinear circuits, on the other hand, is not as good. The best known lower bound is due to a recent result of Alon, Kumar and Volk [AKV20], where they gave an explicit polynomial such that any syntactically multilinear computing it must have size at least $\Omega(n^2/\log^2 n)$. Alon *et al.* expand and improve upon the result by Raz, Shpilka and Yehudayoff [RSY08] who showed an $\Omega(n^{4/3}/\log^2 n)$ lower bound against the same model for the same polynomial.

In chapter 4, we try to give a "reason" as to why showing lower bounds against multilinear circuits might be hard. We do so by showing that weak lower bounds against multilinear circuits can be used to show strong lower bounds against another well studied model of computation, namely *non-commutative* circuits.

Formally, we show the following.

Theorem 1.2.3. *For any $\varepsilon > 0$, assume that there exists an explicit n-variate commutative multilinear polynomial of degree $\mathrm{poly}(n)$, such that any multilinear circuit computing it requires size $\Omega(n^{3+(\omega/2)+\varepsilon})$. Then for any $c > 1$, there exists another explicit m-variate polynomial of degree $\mathrm{poly}(m)$, such that any non-commutative circuit computing it requires size $\Omega(m^c)$.*

Non-Commutative Models of Computation

In the non-commutative setting, the order of multiplication is crucial and therefore, atleast intuitively, the task of proving lower bounds becomes easier in this setting. And indeed, more is known in this setting as compared to the commutative setting.

Even though the best lower bound known against circuits in this setting is still the barely super linear bound by Baur Strassen [Str73a; BS83], which was shown for the general case, in the case of formulas we know exponential lower bounds due to Nisan [Nis91]. In his work, Nisan showed that any formula computing the non-commutative determinant or permanent polynomial must have size $2^{\Omega(n)}$, thus giving an exponential separation between circuits and formulas in this setting.

The proof, in fact, proceeds via a characterisation for the size of the smallest algebraic branching program computing a given non-commutative polynomial, and then giving a lower bound on the size of any ABP computing the non-commutative determinant. Therefore a natural question to ask at this point is whether there is a super-polynomial separation between the powers of formulas and ABPs in the non-commutative setting.

We address this question in chapter 5, where we make progress towards its resolution by showing a tight super-polynomial separation between ABPs and some structured formulas, which we call syntactically abecedarian formulas.

Note that in the non-commutative setting, any monomial in a polynomial can be thought of as a string over the underlying variables. With that in mind, abecedarian polynomials can be defined succinctly using the notion of *regular expressions* from Automata Theory. For a non-commutative polynomial $f \in \mathbb{F}\langle x_1, \ldots, x_n \rangle$, suppose the variables can been partitioned into buckets $\{X_1, \ldots, X_m\}$. Then f is said to be abecedarian with respect to the partition $\{X_1, \ldots, X_m\}$ if the monomials in it are words that can be generated using the regular expression $X_1^* \cdots X_m^*$.

Syntactically abecedarian formulas are simply formulas that have some additional syntactic restrictions that compels them to compute abecedarian polynomials. We

construct a n-variate degree d polynomial that can be computed by efficient ABPs but cannot be computed by syntactically abecedarian formulas of size $n^{o(\log d)}$.

The formal statement is as follows.

Theorem 1.2.4. *Define*

$$\text{linked\_CHSYM}_{n,d}(\mathbf{x}) = \sum_{i_0=1}^{n} \left(\sum_{i_0 \le i_1 \le \ldots \le i_d \le n} x_{i_0,i_1} \cdot x_{i_1,i_2} \cdots x_{i_{d-1},i_d} \right)$$

to be the linked *complete homogeneous polynomial over n-variables of degree d.*

The polynomial $\text{linked\_CHSYM}_{n,d}(\mathbf{x})$ *is* abecedarian *with respect to the partition* $\{X_i : i \in [n]\}$ *where* $X_i = \{x_{i,j} : i \le j \le n\}$. *With respect to this partition,*

1. $\text{linked\_CHSYM}_{n,d}(\mathbf{x})$ *has an* abecedarian *ABP of size* $O(nd)$*;*
2. *any syntactically* abecedarian *formula computing* $\text{linked\_CHSYM}_{n/2,\log n}(\mathbf{x})$ *has size at least* $n^{\Omega(\log \log n)}$.

Recently, Limaye, Srinivasan and Tavenas [LST21a] also made progress towards Nisan's question by showing a super-polynomial separation between ABPs and *homogeneous* formulas. Their work is incomparable to the one presented here since syntactically abecedarian formulas need not be homogeneous.

Constant Depth Circuits

Even though we do not study this restriction in this thesis directly, we do give a summary of some major results for constant depth algebraic circuits since any write up on lower bounds in algebraic circuit complexity would be incomplete if it did not mention them. This is because, unlike the boolean setting, extremely strong depth reduction statements are known in the algebraic world.

The simplest form of such a statement is for formulas. Brent [Bre74] showed that any n-variate, degree d polynomial that is computable by a formula of size s is also computable by a formula of size $\text{poly}(s, n, d)$ and depth $O(\log s)$. A similar statement is also known in the boolean setting due to Spira [Spi71].

Applying techniques similar to those by Brent, Hyafil [Hya79] showed that any polynomial computed by a size s circuit can be equivalently computed by a circuit of depth $O(\log d)$ and size $s^{O(\log d)}$. This was then improved by Valiant, Skyum,

Berkowitz and Rackoff [Val+83], who showed that any n-variate degree d polynomial that can be computed by a circuit of size s can also be computed by a circuit of depth $O(\log d)$ and size $\mathrm{poly}(s, n, d)$. This implied that proving super-polynomial lower bounds for $O(\log d)$ depth circuits is sufficient to prove super-polynomial lower bounds for general arithmetic circuits.

Agrawal and Vinay [AV08] further strengthened this to obtain a depth reduction to depth-4 circuits. They showed that any n-variate degree d polynomial that can be computed by a $2^{o(n)}$ sized circuit can be equivalently computed by homogeneous depth 4 circuit of size $2^{o(n)}$. Their result was strengthened by Koiran [Koi12] and Tavenas [Tav15] to show that any circuit of size s that computes an n-variate degree d polynomial can be computed by a homogeneous depth 4 circuit of size $s^{O(\sqrt{d})}$, and in fact the resulting depth 4 circuits have all multiplication fan-ins bounded by $O(\sqrt{d})$. These results hold over all fields.

Over fields of characteristic zero, Gupta, Kamath, Kayal and Saptharishi [Gup+16] showed that any n-variate degree d polynomial computed by a size s circuit can be equivalently computed by a non-homogeneous depth-3 circuit of size $s^{O(\sqrt{d})}$. Thus, these results formally show that proving good enough lower bounds on circuits of bounded depth is sufficient for proving lower bounds for general circuits.

Therefore much of the focus on proving lower bounds for algebraic circuits shifted towards proving bounds for constant depth circuits [NW97; GK98; GR00; SW01; Raz08; Gup+14; KS15; Fou+15; KS16; Kay+17; CM17; KS17b; KS19; GST20], culminating to a very recent breakthrough result by Limaye, Srinivasan and Tavenas [LST21b] where the showed the first super-polynomial lower bound against any constant depth over fields of characterisic zero. They show that there is an explicit n-variate polynomial which can be computed by efficient algebraic circuits but has no algebraic circuits of depth Δ and size at most $n^{d^{\exp(-O(\Delta))}}$. In fact, for $\Delta = 3, 4$, their lower bound of $n^{\Omega(\sqrt{d})}$ matches the upper bound given by the depth reduction results [Koi12; Tav15; Gup+16].

Organisation of Part I

There are four chapters in the first part of the thesis, where we describe our work on lower bounds in greater detail. The first two chapters (chapter 2, chapter 3) are based on work done with Mrinal Kumar (IIT Bombay), Adrian She (University of Toronto) and Ben Lee Volk (IDC Herzliya), where we show a quadratic lower bound against algebraic branching programs and formulas. The work appears in

the proceedings of CCC 2020 [Cha+20] and the journal version will appear in Computational Complexity.

Following that, in chapter 4, we give a detailed introduction to non-commutative models of computation. Then in chapter 5 we show a tight lower bound against some structured non-commutative formulas and in the process give a super-polynomial separation between the powers of ABPs and these structured formulas. The work appears in the proceedings of CCC 2021 [Cha21].

1.3 Algebraic Independence and Faithful Maps

In the second part of the thesis, we focus on questions related to the notion algebraic independence. Recall that a set of polynomials $\mathbf{f} = \{f_1, \ldots, f_m\} \subset \mathbb{F}[\mathbf{x}]$ is said to be *algebraically dependent* if and only if there is some nonzero polynomial combination of $\{f_1, \ldots, f_m\}$ that is zero. Such a nonzero polynomial $A(z_1, \ldots, z_m) \in \mathbb{F}[\mathbf{z}]$, if one exist, for which $A(f_1, \ldots, f_m) = 0$ is called the *annihilating polynomial* for the set $\{f_1, \ldots, f_m\}$.

We note that the underlying field is very important. For example the polynomials $x + y$ and $x^p + y^p$ are algebraically dependent over $\mathbb{F}_p$ but algebraically independent over a characteristic zero field like $\mathbb{R}$ or $\mathbb{C}$.

Algebraic independence is very well-studied and it is known that algebraically independent subsets of a given set of polynomials form a *matroid* (see [Oxl92]). Hence the size of the maximum algebraically independent subset of $\mathbf{f}$ is well-defined and is called the *algebraic rank* or *transcendence degree* of $\mathbf{f}$. We denote it by $\mathsf{algrank}(\mathbf{f}) = \mathsf{algrank}(f_1, \ldots, f_m)$. A natural question therefore is whether given a set of polynomials, there exists an efficient algorithm to compute their algebraic rank.

One natural approach could be to find the annihilating polynomial of the given set of polynomials. However, it turns out that even checking if the constant term of the annihilating polynomials is zero or not is NP-hard ([Kay09; GSS19]). This effectively rules out any attempt to compute the algebraic rank via properties of the annihilating polynomials. Despite this, over fields of characteristic zero, algebraic rank has an alternate characterisation via the Jacobian criterion.

Jacobi [Jac41] showed that the algebraic rank of a set of polynomials $\mathbf{f}(\subseteq \mathbb{F}[\mathbf{x}])$ is given by the linear rank (over the rational function field $\mathbb{F}(\mathbf{x})$) of the Jacobian of these polynomials. This immediately yields a randomized polynomial time algorithm

to compute the algebraic rank of a given set of polynomials by computing the rank of the Jacobian evaluated at a random point [Ore22; Sch80; Zip79; DL78].

Algebraic Independence over Finite Characteristic. The Jacobian criterion, however, does not continue to hold over fields of finite characteristic unless the polynomials are all quadratics and the underlying field is not of characteristic two[2]. A standard example to exhibit the failure of the Jacobian criterion over fields of characteristic p is $\{x^{p-1}y, y^{p-1}x\}$ — these polynomials are algebraically independent over $\mathbb{F}_p$ but the Jacobian is *not* full-rank over $\mathbb{F}_p$. Pandey, Saxena and Sinhababu [PSS18] characterised the extent of failure of the Jacobian criterion for $\{f_1, \ldots, f_m\}$ by a notion called the *inseparable degree* (Definition 6.3.4) associated with this set.

Over characteristic zero fields, the inseparable degree is always 1 but over fields of characteristic p this is a power of p. In their work, Pandey *et al.* presented a Jacobian-like criterion to capture algebraic independence. Informally, each row of the *generalized Jacobian matrix* is obtained by taking the Taylor expansion of $f_i(\mathbf{x}+\mathbf{z})$ about a generic point, and truncating to just the terms of degree up to the *inseparable degree*. Since the inseparable degree is 1 over characteristic zero, this is just the vector of first order partial derivatives and therefore it corresponds to the *Jacobian matrix*. The formal statement is as follows.

Theorem 1.3.1 ([PSS18])**.** *Let $\{f_1, \ldots f_k\}$ be a set of n-variate polynomials over a field $\mathbb{F}$ with inseparable degree t. Also, for a generic point $\mathbf{z}$, let $\mathcal{H}_t(f_i) = \deg_{\leq t}(f_i(\mathbf{x}+\mathbf{z}) - f_i(\mathbf{z}))$. Then, they are algebraically dependent if and only if there exists a non-zero vector $(\alpha_1, \ldots, \alpha_k) \in \mathbb{F}(\mathbf{z})^k$ such that*

$$\sum_{i=1}^{k} \alpha_i \cdot \mathcal{H}_t(f_i) = 0 \mod \langle \mathcal{H}_t(f_1), \ldots, \mathcal{H}_t(f_k) \rangle^{\geq 2}_{\mathbb{F}(\mathbf{z})} + \langle \mathbf{x} \rangle^{t+1}.$$

In the setting when the *inseparable degree* is constant, this characterisation yields a randomized polynomial time algorithm to compute the algebraic rank.

Whether there exists a randomised polynomial time algorithm to compute the algebraic rank when the inseparable degree is non-constant continues to remain an outstanding open problem. As mentioned earlier, apart from being interesting in its own right, this question also has a connection with the problem of Polynomial Identity Testing via the concept of *faithful homomorphisms*.

[2]Unpublished joint work with Abhibhav Garg, Nitin Saxena and Ramprasad Saptharishi while Ramprasad and I were visiting IIT Kanpur.

Faithful Homomorphisms and Polynomial Identity Testing

Algebraic independence shares a lot of similarities with linear independence due to the matroid structure. One natural task is to find a *rank-preserving transformation* in this setting. This is defined by what are called *faithful homomorphisms*.

Definition 1.3.2 ([BMS13])**.** *Let* $\mathbf{f} = \{f_1, \ldots, f_m\} \subseteq \mathbb{F}[\mathbf{x}]$ *be a set of polynomials. If* $\mathbb{K}$ *is an extension field of* $\mathbb{F}$*, a homomorphism* $\Phi : \mathbb{F}[\mathbf{x}] \to \mathbb{K}[\mathbf{y}]$ *is said to be an* $\mathbb{F}$-faithful *homomorphism for* $\{f_1, \ldots, f_m\}$ *if*

$$\mathsf{algrank}_{\mathbb{F}}\{f_1, \ldots, f_m\} = \mathsf{algrank}_{\mathbb{F}}\{\Phi(f_1), \ldots, \Phi(f_m)\}. \qquad \Diamond$$

Ideally, we would like a faithful homomorphism with $|\mathbf{y}| \approx \mathsf{algrank}\{\mathbf{f}\}$ and $\mathbb{K} = \mathbb{F}$. Beecken, Mittmann and Saxena [BMS13] showed that a *generic* $\mathbb{F}$-linear homomorphism to $\mathsf{algrank}(\mathbf{f})$ many variables would be an $\mathbb{F}$-faithful homomorphism with high probability. One important consequence of faithful homomorphisms is that they preserve nonzeroness of any polynomial composition of $f_1, \ldots, f_m$.

Lemma 1.3.3 ([BMS13; Agr+16])**.** *Suppose* $f_1, \ldots, f_m \in \mathbb{F}[x_1, \ldots, x_n]$ *and* Φ *is an* $\mathbb{F}$*-faithful homomorphism for* $\{f_1, \ldots, f_m\}$*. Then, for any circuit* $C(z_1, \ldots, z_m) \in \mathbb{F}[z_1, \ldots, z_m]$*, we have*

$$C(f_1, \ldots, f_m) = 0 \Leftrightarrow C(\Phi(f_1), \ldots, \Phi(f_m)) = 0.$$

Thus constructing explicit faithful homomorphisms can also be used for polynomial identity testing (PIT), which is the task of checking if a given algebraic circuit C computes the identically zero polynomial. For PIT, the goal is to design a deterministic algorithm that runs in time polynomial in the size of the circuit.

There are two types of PIT algorithms, *whitebox* and *blackbox*. In the blackbox setting, we are only provided evaluation access to the circuit and some of its parameters (such as degree, number of variables, size etc.). Thus blackbox PIT algorithms for a class $\mathcal{C}$ is equivalent to constructing a *hitting set*, which is a list of points in $S \subset \mathbb{F}^n$ such that any nonzero polynomial $f \in \mathcal{C}$ is guaranteed to evaluate to a nonzero value on some $\mathbf{a} \in S$. In the whitebox setting we are additionally provided access to the internal structure of the circuit to carry out the same task.

It follows from Lemma 1.3.3 that if we can construct explicit $\mathbb{F}$-faithful homomorphisms for a set $\{f_1, \ldots, f_m\}$ whose algebraic rank is $k \ll n$, then we have a *variable reduction* that preserves the nonzeroness of any composition $C(f_1, \ldots, f_m)$. This

approach was used by Beecken, Mittmann and Saxena [BMS13] and Agrawal, Saha, Saptharishi, Saxena [Agr+16], in the characteristic zero setting, to design identity tests for several subclasses by constructing faithful maps for $\{f_1, \ldots, f_m\}$ with algebraic rank at most $k = O(1)$, when

- each f_i is a sparse polynomial,
- each f_i is a product of multilinear, variable disjoint, sparse polynomials,
- each f_i is a product of linear polynomials,

and further generalisations.

All the above constructions crucially depend on the fact that the rank of the Jacobian captures algebraic independence. However, this fact is true only over fields of characteristic zero and hence all the above results no longer hold over fields of positive characteristic. Thus, a natural question is whether the criterion of Pandey, Saxena and Sinhababu ([PSS18]) can be used to construct faithful homomorphisms for similar classes of polynomials over fields of finite characteristic.

We answer this question in the affirmative for some restricted settings.

Theorem 1.3.4. *Let $f_1, \ldots, f_m \in \mathbb{F}[x_1, \ldots, x_n]$ be such that* algrank $\{f_1, \ldots, f_m\} = k$ *and the inseparable degree is t. If t and k are bounded by a constant, then we can construct a polynomial (in the input length) sized list of homomorphisms of the form $\Phi : \mathbb{F}[\mathbf{x}] \to \mathbb{F}(s)[y_0, y_1, \ldots, y_k]$ such that at least one of them is guaranteed to be $\mathbb{F}$-faithful for the set $\{f_1, \ldots, f_m\}$, in the following two settings:*

- *When each of the f_i's are sparse polynomials,*
- *When each of the f_i's are products of variable disjoint, multilinear, sparse polynomials.*

Prior to this, construction of faithful homomorphisms over finite fields was known only in the setting when each f_i has small individual degree [BMS13]. Over characteristic zero fields, the inseparable degree is always 1 and hence the faithful maps constructed in [BMS13], [Agr+16] over such fields can be viewed as special cases of our constructions.

The above theorem also holds for a few other models studied by Agrawal *et al.* [Agr+16] (for instance, occur-k products of sparse polynomials). We mention the above two models just as an illustration of lifting the recipe for faithful maps from [BMS13; Agr+16] to the finite characteristic setting. As corollaries, we get efficient PIT algorithms for these models.

Organisation of Part II

We begin with a detailed survey of the results known with respect to efficiently testing algebraic independence of a given set of polynomials in chapter 6. We also show an efficient randomised algorithm to test algebraic independence of quadratic polynomials over fields of odd characteristic. This is an unpublished joint work with Abhibhav Garg (University of Waterloo), Nitin Saxena (IIT Kanpur) and Ramprasad Saptharishi (TIFR Mumbai).

We then, in chapter 7, study its connections to polynomial identity testing and finally give a detailed proof of our work on constructing faithful homomorphisms in various restricted settings thereby giving deterministic algorithms for PIT in certain related settings. This is joint work with Ramprasad Saptharishi (TIFR Mumbai) and the work appears in the proceedings of FSTTCS 2019 [CS19].

Part I

Lower Bounds Against Some Algebraic Models of Computation

2 Lower Bounds Against General Algebraic Branching Programs

In this chapter, we study the model of *Algebraic Branching Programs* (ABPs, for short). They are an intermediate model between algebraic formulas and algebraic circuits. To within polynomial factors, algebraic formulas can be simulated by ABPs, and ABPs can be simulated by circuits. It is believed that each of the reverse transformations requires a super-polynomial blow-up in the size (for some restricted models of computation, this is a known fact [Nis91; Raz06; RY08; Dvi+12; HY16]).

Polynomial families that can be efficiently computed by algebraic branching programs form the complexity class VBP, and the determinant is a complete polynomial for this class under an appropriate notion of reductions. Thus, the famous Permanent vs. Determinant problem is in fact equivalent to showing super-polynomial lower bound for ABPs. Here we focus on the question of proving lower bounds on the size of algebraic branching programs for explicit polynomials.

2.1 Algebraic Branching Programs

We begin by formally defining an algebraic branching program.

Definition 2.1.1 (Algebraic Branching Programs)**.** *An Algebraic Branching Program (ABP) is a layered graph where each edge is labelled by an affine linear form in the underlying variables $\{x_1, \ldots, x_n\}$ and the first and the last layer have one vertex each, called the "start" and the "end" vertex respectively.*

The polynomial computed by an ABP is equal to the sum of the weights of all paths from the start vertex to the end vertex in the ABP, where the weight of a path is equal to the product of the labels of all the edges on it.

The size of an ABP is the number of vertices in it. ◊

While Definition 2.1.1 is quite standard, there are some small variants of it in the literature which we now discuss. These distinctions make no difference as far as

super-polynomial lower bounds are concerned, since each variant can be simulated by the other to within polynomial factors, and thus the issues described here are usually left unaddressed. However, it seems that we are far from proving super-polynomial lower bounds for general algebraic branching programs, and in this paper we focus on proving polynomial (yet still super-linear) lower bounds. In this setting, these distinctions do matter.

Layered vs. Unlayered. In Definition 2.1.1, we have required the graph to be layered. We also consider in this paper ABPs whose underlying graphs are unlayered, which we call *unlayered ABPs*. We are able to prove super-linear lower bounds for this model as well.

One motivation for considering layered graph as the "standard" model is given by the following interpretation. From the definition, it can be observed that any polynomial computable by an ABP with d layers and ℓ_i vertices in the i-th layer can be written as the (only) entry of the 1×1 matrix given by the product $M := \prod_{i=1}^{d-1} M_i$, where M_i is an $\ell_i \times \ell_{i+1}$ matrix with affine forms as entries. One natural complexity measure of such a representation is the total number of non-zero entries in those matrices, which is the number of edges in the ABP. Another natural measure, which can only be smaller, is the sums of dimensions of the matrices involved in the product, which is the same as the number of vertices in the underlying graph.

Branching programs are also prevalent in boolean complexity theory, and in particular in the context of derandomising the class RL. In this setting again it only makes sense to talk about layered graphs.

Unlayered ABPs can also be thought of as (a slight generalization of) *skew circuits*. These are circuits in which on every multiplication gate, at least one of the operands is a variable (or more generally, a linear function).

Edge labels. In Definition 2.1.1 we have allowed each edge label to be an arbitrary affine linear form in the variables. This is again quite standard, perhaps inspired by the characterization due to [Nis91], of the ABP complexity of a non-commutative polynomial as the rank of an associated coefficients matrix.

A more restrictive definition would only allow each edge to be labelled by a linear function in 1 variable. On the other hand, an even more general definition, which we sometimes adopt, is to allow every edge to be labelled by an *arbitrary* polynomial of degree at most Δ. In this case we refer to the model as an ABP with edge labels of degree at most Δ. Thus, the common case is $\Delta = 1$, but our results are meaningful

even when $\Delta = \omega(1)$. Note that this is quite a powerful model, which is allowed to use polynomials with super-polynomial standard circuit complexity "for free".

We will recall some of these distinctions in section 3.1, where we discuss previous results and some of these apply to several of the variants discussed here.

2.2 Our Results

Our first result is a quadratic lower bound on the size of any algebraic branching program computing some explicit polynomial.

Theorem 2.2.1. *Let $\mathbb{F}$ be a field and $n \in \mathbb{N}$ such that $\mathrm{char}(\mathbb{F}) \nmid n$. Then any algebraic branching program over $\mathbb{F}$ computing the polynomial $\sum_{i=1}^{n} x_i^n$ is of size $\Omega(n^2)$. When the ABP's edge labels are allowed to be polynomials of degree at most Δ, our lower bound is $\Omega(n^2/\Delta)$.*

Note that there also exists an algebraic branching program for $\sum_{i=1}^{n} x_i^n$ of size $O(n^2/\Delta)$, which shows that our bound is in fact tight.

A rough sketch of the construction is as follows. The ABP essentially consists of n parallel paths from the source vertex to the target vertex, with the i^{th} path computing x_i^n. If the labels on the edges are allowed to have degree $\leq \Delta$, then each path consists of n/Δ edges ($\lceil n/\Delta \rceil$ to be more precise), with all the edges on the i^{th} path being labelled by x_i^{Δ} (except possibly the last edge, which is labelled by $x_i^{n-\Delta((n/\Delta)-1)}$).

For the unlayered case, we prove a weaker superlinear lower bound on the number of edges. Note that this is weaker than proving a lower bound on the size (defined to be the number of vertices).

Theorem 2.2.2. *Let $\mathbb{F}$ be a field and $n \in \mathbb{N}$ such that $\mathrm{char}(\mathbb{F}) \nmid n$. Then any unlayered algebraic branching program over $\mathbb{F}$ that computes the polynomial $\sum_{i=1}^{n} x_i^n$ and has edges labels of degree at most Δ must have at least $\Omega(n \log n/(\log \log n + \log \Delta))$ edges.*

Previous Work

The best lower bound known for ABPs prior to this work is a lower bound of $\Omega(n \log n)$ on the number of edges for the same polynomial $\sum_{i=1}^{n} x_i^n$. This follows from the classical lower bound of $\Omega(n \log n)$ by [Str73a; BS83] on the number of

multiplication gates in any algebraic circuit computing the polynomial $\sum_{i=1}^{n} x_i^n$ and the observation that when converting an ABP to an algebraic circuit, the number of product gates in the resulting circuit is at most the number of edges in the ABP.

Theorem 2.2.1 improves upon this bound quantitatively, and also qualitatively, since the lower bound is on the number of vertices in the ABP. Further, it is also not hard to show that $\sum_{i=1}^{n} x_i^n$ can be computed by algebraic circuits of size $O(n \log n)$. Therefore, Theorem 2.2.1 gives a super-linear separation between the relative powers of algebraic circuits and ABPs.

For the case of homogeneous ABPs[1] however, a quadratic lower bound for the polynomial $\sum_{i=1}^{n} x_i^n$ was shown by [Kum19]. In a nutshell, the Kumar's result is equivalent to a lower bound for ABPs computing the polynomial $\sum_{i=1}^{n} x_i^n$ when the number of layers in the ABP is at most n. In this work, we generalize this to proving essentially the same lower bound for ABPs with an unbounded number of layers.

In general, an ABP computing an n-variate homogeneous polynomial of degree $\mathrm{poly}(n)$ can be homogenized with a polynomial blow-up in size. This is proved in a similar manner to the standard classical result of [Str73b] which shows this statement for algebraic circuits. Thus, much like the discussion following Definition 2.1.1, homogeneity is not an issue when one considers polynomial vs. super-polynomial sizes, but becomes relevant when proving polynomial lower bounds. In other contexts in algebraic complexity this distinction is even more sharp. For example, exponential lower bounds for homogeneous depth-3 circuits are well known due to [NW97], but strong enough exponential lower bounds for non-homogeneous depth-3 circuits would separate VP from VNP (as shown by [Gup+16]).

For *unlayered* ABPs, the situation is more complex. If the edge labels are only functions of one variable, then we need to only consider multilinear polynomials for the lower bound to be non-trivial. It is possible to adapt the techniques of [Nec66] in order to obtain a lower bound of $\tilde{\Omega}(n^{3/2})$ for a multilinear polynomial in this model. This is an argument attributed to Pudlák and sketched by [KW93] for the boolean model of parity branching programs, but can be applied to the algebraic setting. However, this argument does not extend to the case where the edge labels are arbitrary linear or low-degree polynomials in the n variables. The crux of Nechiporuk's argument is to partition the variables into m disjoint sets, to argue (using counting or dimension arguments) that the number of edges labelled by

[1] An ABP is *homogeneous* if the polynomial computed between the start vertex and any other vertex is a homogeneous polynomial. This condition is essentially equivalent to assuming that the number of layers in the ABP is upper bounded by the degree of the output polynomial.

variables from each set must be somewhat large,[2] and then to sum the contributions over all m sets. This is hard to implement in models where a single edge can have a "global" access to all variables, since it is not clear how to avoid over-counting in this case.

As mentioned above, the lower bound of [BS83; Str73a] does hold in the unlayered case, assuming the edge labels are linear functions in the variables. However, when we allow edge labels of degree at most Δ for some $\Delta \geq 2$, their technique does not seem to carry over. Indeed, a key step in the Baur-Strassen proof is the claim that if a polynomial f has a circuit of size τ, then there is a circuit of size $O(\tau)$ which computes all its first order partial derivatives. This statement does not seem to hold if we equip the circuit with the ability to compute such low-degree polynomials "for free".

By suitably extending the techniques of [Ben83; Ben94], however, it is possible to get an $\Omega(n \log n / \log \Delta)$ lower bound for this model for a different polynomial. Our lower bounds are weaker by at most a doubly-logarithmic factor; however, the techniques are completely different. Ben-Or's proofs rely on strong modern results in algebraic geometry, whereas our proofs are elementary.

Proof Overview

The first part in the proof of Theorem 2.2.1 is an extension of the lower bound proved by [Kum19] for ABPs with at most n layers. This straightforward but conceptually important adaptation shows that a similar lower bound holds for any polynomial of the form

$$\sum_{i=1}^{n} x_i^n + \varepsilon(\mathbf{x}) \,,$$

where the suggestively named $\varepsilon(\mathbf{x})$ should be thought of as an "error term" which is "negligible" as far as the proof in [Kum19] is concerned. The exact structure we require is that $\varepsilon(\mathbf{x})$ is of the form $\sum_{i=1}^{r} P_i Q_i + R$, where P_i, Q_i are polynomials with no constant term and $\deg(R) \leq n - 1$. The parameter r measures the "size" of the error, which we want to keep small, and the lower bound holds if, for example, $r \leq n/10$.

[2] This is usually guaranteed by constructing a function or a polynomial with the property that given a fixed set S in the partition, there are many subfunctions or subpolynomials on the variables of S that can be obtained by different restrictions of the variables outside of S.

To argue about ABPs with d layers, with $d > n$, we use a notion of depth reduction which is reminiscent of similar statements in the context of matrix rigidity. We show that unless the size τ of the ABP is too large to begin with (in which case there is nothing to prove), it is possible to find a small set of vertices (of size about $\eta = \tau/d$) whose removal adds a small error term $\varepsilon(\mathbf{x})$ as above with at most η summands, but also reduces the depth of the ABP by a constant factor. Repeatedly applying this operation $O(\log n)$ times eventually gives an ABP of depth at most n while ensuring that we have not accumulated too much "error",[3] so that we can apply the lower bound from the previous paragraph. In the full proof we have to be a bit more careful when arguing about the ABP along the steps of the proof above. The details are presented in section 2.4.

The proof of Theorem 2.2.2 follows the same strategy, although the main impediment is that general undirected graphs can have much more complex structure than layered graphs. One of the main ingredients in our proof is (a small variant of) a lemma of [Val77], which shows that every graph of depth 2^k with m edges contains a set of $< m/k$ edges whose removal reduces the depth of the graph to 2^{k-1}. This lemma helps us identify a small set of vertices which can reduce the depth of the graph by a constant factor while again accumulating small error terms.

Interestingly, Valiant originally proved this lemma in a different context, where he showed that linear algebraic circuits of depth $O(\log n)$ and size $O(n)$ can be reduced to a special type of depth-2 circuits (and thus strong lower bounds on such circuits imply super-linear lower bounds on circuits of depth $O(\log n)$). This lemma can be also used to show that *boolean* circuits of depth $O(\log n)$ and size $O(n)$ can be converted to depth-3 circuits of size $2^{o(n)}$, and thus again strong lower bounds on depth-3 circuits will imply super-linear lower bounds on circuits of depth $O(\log n)$. Both of these questions continue to be well known open problems in algebraic and boolean complexity, and to the best of our knowledge, our proof is the first time Valiant's lemma is successfully used in order to prove circuit lower bounds for explicit functions or polynomials.

Elementary Symmetric vs Power Symmetric Polynomials. We remark that the lower bound in Theorem 2.4.1 also holds for elementary symmetric polynomials of degree $0.1n$ on n variables. However, the proof seems a bit simpler and more instructive for the case of power symmetric polynomials and so we state and prove

[3] It takes some care in showing that the total number of error terms accumulated is at most $n/10$ as opposed to the obvious upper bound of $O(n \log n)$. In particular, we observe that the number of error terms can be upper bounded by a geometric progression with first term roughly τ/n and common ratio being a constant less than 1.

the theorem for these polynomials. In general, these lower bounds hold for any family of polynomials of high enough degree whose zeroes of multiplicity at least two lie in a low dimensional variety, or more formally, an analogue of Claim 2.4.7 is true. In particular, such a statement is true for the elementary symmetric polynomial (see, for example, Lemma 3.2.5).

2.3 Preliminaries

All logarithms in this chapter are base 2.

We use some standard graph theory terminology: If G is a directed graph and (u, v) is an edge, v is called the *head* of the edge and u the *tail*. The directed graphs studied here are always acyclic with a designated source vertex s and a sink vertex t. The *depth* of a vertex v, denoted $\mathsf{depth}(v)$, is the length (in edges) of a longest path from s to v. The depth of the graph, denoted by $\mathsf{depth}(G)$, is the depth of t.

For any two vertices u and v in an ABP, the polynomial computed between u and v is the sum of weights of all paths between u and v in the ABP. We denote this by $[u, v]$.

The formal degree of a vertex u in an ABP denoted $\mathrm{fdeg}(u)$, is defined inductively as follows: If s is the start vertex of the ABP, $\mathrm{fdeg}(s) = 0$. If u is a vertex with incoming edges from $u_1, \ldots, u_k$, labelled by non-zero polynomials $\ell_1, \ldots, \ell_k$, respectively, then

$$\mathrm{fdeg}(u) = \max_{i \in [k]} \{\deg(\ell_i) + \mathrm{fdeg}(u_i)\}\,.$$

It follows by induction that for every vertex u, $\deg([s, u]) \leq \mathrm{fdeg}(u)$ (however, cancellations can allow for arbitrary gaps between the two). Also, note that the formal degree of vertices is monotonically non-decreasing along any path from the source vertex to the sink vertex. The formal degree of the ABP is the formal degree of the sink vertex.

We sometimes denote by $\mathbf{x}$ the vector of variables $(x_1, \ldots, x_n)$, where n is understood from the context. Similarly we use $\mathbf{0}$ to denote the n-dimensional vector $(0, 0, \ldots, 0)$.

Variety and its Dimension

An important notion we will use in our proofs is that of an affine algebraic variety. Given a set of polynomials, the algebraic variety defined by these polynomials is defined to be the set of their common zeros. That is, if S is a set of polynomials on n variables over a field $\mathbb{F}$, then

$$\mathbb{V}(S) = \{\mathbf{a} \in \mathbb{F}^n \ : \ \forall f \in S, f(\mathbf{a}) = 0\}\,.$$

Given a variety, an important property that is studied is its dimension. Intuitively, it is an appropriate generalisation of the notion of dimension for linear spaces and roughly captures the degrees of freedom of the variety.

We will not be defining it formally here and refer the interested reader to the book by Cox, Little and O'Shea [CLO07]. However we state formally the properties, of dimensions of varieties, that we will be using in our proofs.

Lemma 2.3.1 (Section 2.8 in [Smi14])**.** *Let S be a set of polynomials in n variables over an algebraically closed field $\mathbb{F}$ such that $|S| \leq n$. Let $V = \mathbb{V}(S)$ be the set of common zeros of polynomials in S. If V is non-empty, then the dimension of V is at least $n - |S|$.*

Lemma 2.3.2 (Chapter 4 in [CLO07])**.** *Let $\mathbb{F}$ be an algebraically closed field, and let $V_1 \subseteq \mathbb{F}^n$ and $V_2 \subseteq \mathbb{F}^n$ be two affine varieties such that $V_1 \subseteq V_2$. Then, the dimension of V_1 is at most the dimension of V_2.*

We now move on to proving our main theorems. In section 2.4 we prove Theorem 2.4.1 whereas in section 2.5 we prove Theorem 2.2.2.

2.4 A Lower Bound Against ABPs

In this section we prove Theorem 2.2.1. We start by restating it.

Theorem 2.4.1. *Let $\mathbb{F}$ be a field and $n \in \mathbb{N}$ such that $\mathrm{char}(\mathbb{F}) \nmid n$. Then any algebraic branching program over $\mathbb{F}$ computing the polynomial $\sum_{i=1}^{n} x_i^n$ is of size at least $\Omega(n^2)$.*

When the ABP's edge labels are allowed to be polynomials of degree at most Δ, our lower bound is $\Omega(n^2/\Delta)$.

For technical reasons, we work with a slightly more general model which we call *multilayered* ABPs, which we now define.

Definition 2.4.2 (Multilayered ABP)**.** *Let $\mathcal{A}_1, \ldots, \mathcal{A}_k$ be k ABPs with $d_1, \ldots, d_k$ layers and $\tau_1, \ldots, \tau_k$ vertices, respectively. A multilayered ABP $\mathcal{A}$, denoted by $\mathcal{A} = \sum_{i=1}^{k} \mathcal{A}_i$, is the ABP obtained by placing $\mathcal{A}_1, \mathcal{A}_2, \ldots, \mathcal{A}_k$ in parallel and identifying their start and end vertices respectively. Thus, the polynomial computed by $\mathcal{A}$ is $\sum_{i=1}^{k} [\mathcal{A}_i]$, where $[\mathcal{A}_i]$ is the polynomial computed by $\mathcal{A}_i$.*

The number of layers of $\mathcal{A}$ is $d := \max\{d_1, \ldots, d_k\}$. The size of $\mathcal{A}$ is the number of vertices in $\mathcal{A}$, and thus equals

$$|\mathcal{A}| := 2 + \sum_i (\tau_i - 2)\,. \qquad \Diamond$$

Note that this model is an intermediate model between (layered) ABPs and unlayered ABPs: given a multilayered ABP of size τ it is straightforward to construct an unlayered ABP of size $O(\tau)$ which computes the same polynomial.

A Decomposition Lemma

The following lemma gives a decomposition of a (possibly unlayered) ABP in terms of the intermediate polynomials it computes. Its proof closely resembles that of Lemma 3.5 in [Kum19]. For completeness we prove it here for a slightly more general model.

Lemma 2.4.3 ([Kum19])**.** *Let $\mathcal{B}$ be a (possibly unlayered) algebraic branching program which computes a degree d polynomial $P \in \mathbb{F}[x_1, \ldots x_n]$, and has formal degree d. Further, assume that the edges of $\mathcal{B}$ are labelled by arbitrary polynomials of degree at most Δ, where $1 \leq \Delta \leq d/2$. Set $d' = \lfloor d/\Delta \rfloor$. For any $i \in \{1, 2, \ldots, d' - 1\}$, let $S_i = \{u_{i,1}, u_{i,2}, \ldots, u_{i,m}\}$ be the set of all vertices in $\mathcal{B}$ whose formal degree is in the interval $[i\Delta, (i+1)\Delta)$.*

Then, there exist polynomials $Q_{i,1}, Q_{i,2}, \ldots, Q_{i,m}$ and R_i, each of degree at most $d - 1$ such that

$$P = \sum_{j=1}^{m} [s, u_{i,j}] \cdot Q_{i,j} + R_i\,.$$

Proof. Fix i as above and set $S_i = \{u_{i,1}, u_{i,2}, \ldots, u_{i,m}\}$ as above (observe that since each edge label is of degree at most Δ, S_i is non empty). Further suppose, without loss of generality, that the elements of S_i are ordered such that there is no directed path from $u_{i,j}$ to $u_{i,j'}$ for $j' > j$.

Consider the unlayered ABP $\mathcal{B}_1$ obtained from $\mathcal{B}$ by erasing all incoming edges to $u_{i,1}$, and multiplying all the labels of the outgoing edges from $u_{i,1}$ by a new variable y_1. The ABP $\mathcal{B}_1$ now computes a polynomial of the form

$$P'(y_1, x_1, \ldots, x_n) = y_1 \cdot Q_{i,1} + R_{i,1}$$

where $P = P'([s, u_{i,1}], x_1, \ldots, x_n)$. $R_{i,1}$ is the polynomial obtained from $\mathcal{B}_1$ by setting y_1 to zero, or equivalently, removing $u_{i,1}$ and all its outgoing edges.

The same step is similarly applied on $\mathcal{B}_1$ with $u_{i,2}$ to obtain $\mathcal{B}_2$ and so on till we obtain the ABP $\mathcal{B}_m$ that computes

$$P = \sum_{j=1}^{m} [s, u_{i,j}] \cdot Q_{i,j} + R_i.$$

(See Figure 2.1 for an illustration of this decomposition). Indeed, observe that since there is no path from $u_{i,j}$ to $u_{i,j'}$ for $j' > j$, removing $u_{i,j}$ does not change $[s, u_{i,j'}]$. The bound on the degrees of $Q_{i,j}$ is immediate from the fact that the formal degree of the ABP is at most d and $\mathrm{fdeg}(u_{i,j}) \geq 1$. It remains to argue the $\deg(R_i) \leq d - 1$.

The polynomial R_i is obtained from $\mathcal{B}$ by erasing all the vertices in S_i and the edges touching them. We will show that every path in the corresponding ABP computes a polynomial of degree at most $d - 1$. Let $s = v_1, v_2, \ldots, v_r = t$ be such a path, which is also a path in $\mathcal{B}$. Let v_k be the minimal vertex in the path whose degree (in $\mathcal{B}$) is at least $(i + 1)\Delta$ (if no such v_k exists, the proposition follows). As $v_{k-1} \notin S_i$, the formal degree of v_{k-1} is at most $i\Delta - 1$. The degree of the polynomial computed by this path is thus at most $i\Delta - 1 + \Delta + D = (i + 1)\Delta - 1 + D$, where D is the degree of product of the labels on the path $v_k, v_{k+1}, \ldots, t$. To complete the proof, it remains to be shown that $D \leq d - (i + 1)\Delta$.

Indeed, if $D \geq d - (i + 1)\Delta + 1$ then since the degree of v_k is at least $(i + 1)\Delta$, there would be in $\mathcal{B}$ a path of formal degree at least $(i + 1)\Delta + D \geq d + 1$, contradicting the assumption on $\mathcal{B}$. □

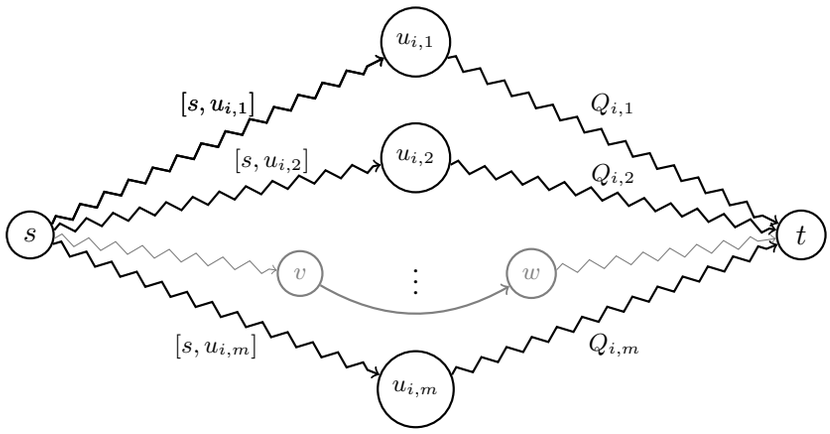


Figure 2.1.: Decomposing an ABP by expanding an intermediate layer as in Lemma 2.4.3. Zigzag edges represent paths in the graph and not necessarily a single edge. The gray path does not pass through any vertex in S_i. Note that paths from $u_{i,j}$ to $u_{i,j'}$ for $j' < j$ are possible, but are not depicted for simplicity.

A Robust Lower Bound for ABPs of Formal Degree At Most n

In this section, we prove a lower bound for the case where the formal degree of every vertex in the ABP is at most n. In fact, [Kum19] has already proved a quadratic lower bound for this case.

Theorem 2.4.4 ([Kum19])**.** *Let $n \in \mathbb{N}$ and let $\mathbb{F}$ be a field such that $\mathrm{char}(\mathbb{F}) \nmid n$. Then any algebraic branching program of formal degree at most n which computes the polynomial $\sum_{i=1}^{n} x_i^n$ has at least $\Omega(n^2)$ vertices.*

However, to prove Theorem 2.4.1, we need the following more "robust" version of Theorem 2.4.4, which gives a lower bound for a larger class of polynomials. For completeness, we also sketch an argument for the proof which is a minor variation of the proof of Theorem 2.4.4.

Theorem 2.4.5. *Let $n \in \mathbb{N}$ and let $\mathbb{F}$ be field such that $\mathrm{char}(\mathbb{F}) \nmid n$. Further, let $A_1(\mathbf{x}), \ldots, A_r(\mathbf{x}), B_1(\mathbf{x}), \ldots, B_r(\mathbf{x})$ and $R(\mathbf{x})$ be polynomials such that for every i, $A_i(\mathbf{0}) = B_i(\mathbf{0}) = 0$ and R is a polynomial of degree at most $n - 1$. Then, any (possibly unlayered) algebraic branching program over $\mathbb{F}$, of formal degree at most n and edge labels of degree at most $\Delta \leq n/10$, which computes the polynomial*

$$\sum_{i=1}^{n} x_i^n + \sum_{j=1}^{r} A_j \cdot B_j + R$$

has at least $\frac{(n/2-r)n}{2\Delta}$ *vertices.*

The proof of the theorem follows from Lemma 2.4.3 and the following lemma which is a slight generalization of [Kum19, Lemma 3.1]. We include a proof for completeness.

Lemma 2.4.6. *Let* $n \in \mathbb{N}$*, and let* $\mathbb{F}$ *be an algebraically closed field such that* $\mathrm{char}(\mathbb{F}) \nmid n$*. Let* $\{P_1, \ldots, P_m, Q_1, \ldots, Q_m\}$ *and* $\{A_1, \ldots, A_r, B_1, \ldots, B_r\}$ *be a set of polynomials in* $\mathbb{F}[x_1, \ldots, x_n]$ *such that the set of their common zeros*

$$V = \mathbb{V}(P_1, \ldots, P_m, Q_1, \ldots, Q_m, A_1, \ldots, A_r, B_1, \ldots, B_r) \subseteq \mathbb{F}^n$$

is non-empty. Finally, suppose R *is a polynomial in* $\mathbb{F}[\mathbf{x}]$ *of degree at most* $n-1$*, such that*

$$\sum_{i=1}^{n} x_i^n + \sum_{j=1}^{r} A_j \cdot B_j = R + \sum_{i=1}^{m} P_i \cdot Q_i.$$

Then, $m \geq \frac{n}{2} - r$*.*

Proof. By Lemma 2.3.1, since $V \neq \emptyset$, $\dim(V) \geq n - 2m - 2r$. Now note that every element of V is also a zeros with multiplicity two of $\sum_{i=1}^{n} x_i^n - R$, since

$$\sum_{i=1}^{n} x_i^n - R = \sum_{i=1}^{m} P_i \cdot Q_i - \sum_{j=1}^{r} A_j \cdot B_j \,.$$

Therefore, the set of zeroes of multiplicity two of $\sum_{i=1}^{n} x_i^n - R$ has dimension at least $n - 2m - 2r$. In other words, if S is the set of common zeros of the set of all first order partial derivatives of $\sum_{i=1}^{n} x_i^n - R$, then $\dim(S) \geq n - 2m - 2r$. Up to scaling by n (which is non-zero in $\mathbb{F}$, by assumption), the set of all first order partial derivatives of $\sum_{i=1}^{n} x_i^n - R$ is given by

$$\left\{ x_i^{n-1} - \frac{1}{n} \partial_{x_i} R \right\}_{i \in [n]} .$$

Thus, the statement of this lemma immediately follows from the following claim.

Claim 2.4.7 (Lemma 3.2 in [Kum19])**.** *Let* $\mathbb{F}$ *be an algebraically closed field, and* D *a positive natural number. For every choice of polynomials* $g_1, g_2, \ldots g_n \in \mathbb{F}[\mathbf{x}]$ *of degree at most* $D-1$*, the dimension of the variety*

$$\mathbb{V}(x_1^D - g_1, x_2^D - g_2, \ldots, x_n^D - g_n)$$

is zero.

Indeed, the above claim shows that $0 = \dim(S) \geq n - 2m - 2r$, and so it must be the case that $m \geq \frac{n}{2} - r$. This completes the proof of Lemma 2.4.6. □

We now use Lemma 2.4.3 and Lemma 2.4.6 to complete the proof of Theorem 2.4.5.

Proof of Theorem 2.4.5. Let $\mathcal{B}$ be an algebraic branching program of formal degree at most n, edge labels of degree at most $\Delta \leq n/10$, and with start vertex s and end vertex t, which computes

$$\sum_{i=1}^{n} x_i^n + \sum_{j=1}^{r} A_j \cdot B_j + R.$$

We may assume without loss of generality that $\mathbb{F}$ is algebraically closed, by interpreting $\mathcal{B}$ as an ABP over the algebraic closure of $\mathbb{F}$, if necessary.

Let $n' = \lfloor n/\Delta \rfloor$ and $k \in \{1, 2, \ldots, n' - 1\}$ be a arbitrarily fixed. Further, let $V_k = \{v_{k,1}, v_{k,2}, \ldots, v_{k,m}\}$ be the set of all vertices in $\mathcal{B}$ whose formal degree lies in the interval $[k\Delta, (k+1)\Delta)$. Letting $P'_j = [s, v_{k,j}]$, by Lemma 2.4.3, there exist polynomials $Q'_1, Q'_2, \ldots, Q'_m$ and R', each of degree at most $n - 1$ such that

$$\sum_{i=1}^{n} x_i^n + \sum_{j=1}^{r} A_j \cdot B_j + R = \sum_{j=1}^{m} P'_j \cdot Q'_j + R' \,.$$

Let α_j, β_j be the constant terms in P'_j, Q'_j respectively. Then by defining

$$P_j = P'_j - \alpha_j \qquad \text{and} \qquad Q_j = Q'_j - \beta_j,$$

we have that

$$\sum_{i=1}^{n} x_i^n + \sum_{j=1}^{r} A_j \cdot B_j = R'' + \sum_{j=1}^{m} P_j \cdot Q_j.$$

Here, $R'' = -R + R' + \sum_{j=1}^{m} (\alpha_j \cdot Q'_j + \beta_j \cdot P'_j + \alpha_j \beta_j)$. We now have that for every i, the constant terms of P_i, Q_i are zero and $\deg(R'') \leq n - 1$. Let

$$\mathcal{V} = \mathbb{V}(P_1, \ldots, P_m, Q_1, \ldots, Q_m, A_1, \ldots, A_r, B_1, \ldots, B_r).$$

Then $\mathbf{0} \in \mathcal{V}$, and so $\mathcal{V} \neq \emptyset$. Thus by Lemma 2.4.6, we know that $m \geq \frac{n}{2} - r$.

Finally, for $k \neq k' \in \{1, 2, \ldots, n' - 1\}$, $V_k \cap V_{k'} = \emptyset$. Thus, the number of vertices in $\mathcal{B}$ must be at least

$$\left(\frac{n}{2} - r\right) \cdot (n' - 1) \geq \left(\frac{n}{2} - r\right) \cdot \frac{n}{2\Delta} . \qquad \square$$

A Lower Bound for The General Case

The following lemma shows how, given an ABP with d layers which computes a polynomial F, we can obtain a multilayered ABP whose number of layers is significantly smaller and computes F plus a small "error term".

Lemma 2.4.8. *Let $\mathcal{A}$ be an ABP over a field $\mathbb{F}$ with d layers that computes the polynomial F and has m vertices. Let s, t be the start, end vertices of $\mathcal{A}$ respectively and let $L = \left\{u_1, u_2, \ldots, u_{|L|}\right\}$ be the set of vertices in the ℓ-th layer of $\mathcal{A}$.*

For every $i \in \{1, 2, \ldots, |L|\}$, let α_i and β_i be the constant terms of $[s, u_i]$ and $[u_i, t]$ respectively. Furthermore, let P_i and Q_i be polynomials such that $[s, u_i] = P_i + \alpha_i$ and $[u_i, t] = Q_i + \beta_i$.

Then there is a multilayered ABP $\mathcal{A}'$ computing the polynomial

$$F - \sum_{i=1}^{|L|} P_i \cdot Q_i + \sum_{i=1}^{|L|} \alpha_i \cdot \beta_i$$

that has size at most $|\mathcal{A}|$ and at most $\max\{\ell, d - \ell + 1\}$ layers.

Figure 2.2 gives an intuition for the proof of this lemma. A formal proof is as follows.

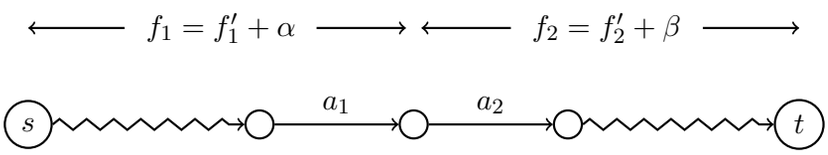
Proof. Let $u_1, u_2, \ldots, u_{|L|}$ be the vertices in L as described, so that

$$F = [s, t] = \sum_{i=1}^{|L|} [s, u_i] \cdot [u_i, t] .$$

Further, for every $i \in \{1, 2, \ldots, |L|\}$, $[s, u_i] = P_i + \alpha_i$ and $[u_i, t] = Q_i + \beta_i$, where the constant terms of P_i and Q_i are zero (by definition). Having set up this notation, we can thus express the polynomial F computed by $\mathcal{A}$ as

$$F = [s, t] = \sum_{i=1}^{|L|} (P_i + \alpha_i) \cdot (Q_i + \beta_i) .$$

$\mathcal{A} = f_1 \cdot f_2$



$\mathcal{A}' = \beta \cdot f_1 + \alpha \cdot f_2 = \mathcal{A}_\ell - f_1' \cdot f_2' + \alpha \cdot \beta$

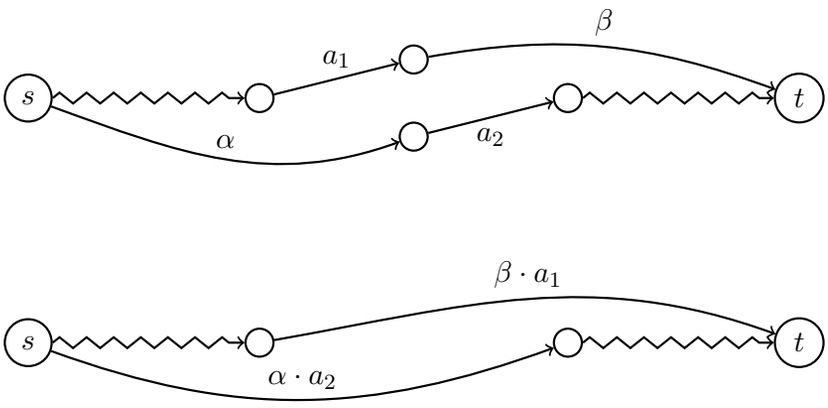


Figure 2.2.: Proof of Lemma 2.4.8 when $|L| = 1$. When $|L|$ is larger, the proof follows similarly.

On further rearrangement, this gives

$$F - \left(\sum_{i=1}^{|L|} P_i \cdot Q_i\right) + \left(\sum_{i=1}^{|L|} \alpha_i \cdot \beta_i\right) = \left(\sum_{i=1}^{|L|} \alpha_i \cdot (Q_i + \beta_i)\right) + \left(\sum_{i=1}^{|L|} (P_i + \alpha_i) \cdot \beta_i\right) .$$

This is equivalent to the following expression.

$$F - \left(\sum_{i=1}^{|L|} P_i \cdot Q_i\right) + \left(\sum_{i=1}^{|L|} \alpha_i \cdot \beta_i\right) = \left(\sum_{i=1}^{|L|} \alpha_i \cdot [u_i, t]\right) + \left(\sum_{i=1}^{|L|} [s, u_i] \cdot \beta_i\right) .$$

Now, observe that the polynomial $\sum_{i=1}^{|L|} [s, u_i] \cdot \beta_i$ is computable by an ABP $\mathcal{B}$ with $\ell + 1$ layers, obtained by just keeping the vertices and edges within first ℓ layers of $\mathcal{A}$ and the end vertex t, deleting all other vertices and edges, and connecting the vertex u_i in the ℓ-th layer to t by an edge of weight β_i. Similarly, the polynomial $\sum_{i=1}^{|L|} \alpha_i \cdot [u_i, t]$ is computable by an ABP $\mathcal{C}$ with at most $(d - \ell + 1) + 1$ layers, whose set of vertices is s along the vertices in the layers $\ell, \ell + 1, \ell + 2, \ldots, d$ of $\mathcal{A}$. From the definition of $\mathcal{B}$ and $\mathcal{C}$, it follows that the multilayered ABP $\tilde{\mathcal{A}}$ obtained by taking the sum of $\mathcal{B}$ and $\mathcal{C}$ has at most $\max\{\ell + 1, d - \ell + 2\}$ layers.

We are almost done with the proof of the lemma, except for the upper bound on the number of vertices of the resulting multilayered ABP $\tilde{\mathcal{A}}$, and the fact that the upper bound on the depth is slightly weaker than claimed. Both these issues can be solved simultaneously.

The vertices in L appear in both the ABP $\mathcal{B}$ and the ABP $\mathcal{C}$ and are counted twice in the size of $\tilde{\mathcal{A}}$. However, every other vertex is counted exactly once. Hence,

$$|\mathcal{B}| + |\mathcal{C}| = |\mathcal{A}| + |L| . \tag{2.4.9}$$

In order to fix this issue, we first observe that the edges between the vertices in the ℓ-th layer of $\mathcal{B}$ and the end vertex t are labelled by $\beta_1, \beta_2, \ldots, \beta_{|L|}$, all of which are field constants. In the following claim, we argue that for ABPs with this additional structure, the last layer is redundant and can be removed.

Claim 2.4.10. *Let $\mathcal{M}$ be an ABP over $\mathbb{F}$ with $k+1$ layers and edge labels of degree at most Δ such that the labels of all the edges between the k-th layer of $\mathcal{M}$ and its end vertex are scalars in $\mathbb{F}$. Then, there is an ABP $\mathcal{M}'$ with k layers computing the same polynomial as $\mathcal{M}$, with edge labels of degree at most Δ, such that*

$$|\mathcal{M}'| \leq |\mathcal{M}| - |V| \ ,$$

where V is the set of vertices in the k-th layer of $\mathcal{M}$.

An analogous statement, with an identical proof, is true if we assume that all edge labels between the first and second layer are scalars in $\mathbb{F}$.

We first use Claim 2.4.10 to complete the proof of the lemma. As observed above, the edge labels between the last layer L of $\mathcal{B}$ and its end vertex are all constants. Hence, by Claim 2.4.10, there is an ABP $\mathcal{B}'$ which computes the same polynomial as $\mathcal{B}$ such that $|\mathcal{B}'| \leq |\mathcal{B}| - |L|$, and $\mathcal{B}'$ has only ℓ layers. Similarly, we can obtain an ABP $\mathcal{C}'$ with at most $d - \ell + 1$ layers.

We consider the multilayered ABP $\mathcal{A}'$ by taking the sum of $\mathcal{B}'$ and $\mathcal{C}'$. Clearly, the number of layers in $\mathcal{A}'$ is at most $\max\{\ell, d-\ell+1\}$ and the size is at most

$$|\mathcal{A}'| \leq |\mathcal{B}'| + |\mathcal{C}'| \leq (|\mathcal{B}| - |L|) + (|\mathcal{C}| - |L|) \leq |\mathcal{A}| \ .$$

Here, the second inequality follows by Claim 2.4.10 and the last one follows by Equation 2.4.9. To complete the proof of the lemma, we now prove Claim 2.4.10. □

Proof of Claim 2.4.10. For the proof of the claim, we focus on the k-th and $(k-1)$-st layer of $\mathcal{M}$. To this end, we first set up some notation. Let $\{v_1, v_2, \ldots, v_r\}$ be the set of vertices in the k-th layer of $\mathcal{M}$, $\{u_1, u_2, \ldots, u_{r'}\}$ be the set of vertices in $(k-1)$-st layer of $\mathcal{M}$, and a, b denote the start and the end vertices of $\mathcal{M}$ respectively. Then, the polynomial computed by $\mathcal{M}$, can be decomposed as

$$[a, b] = \sum_{i=1}^{r} [a, v_i] \cdot [v_i, b] \ .$$

Note that (v_i, b) is an edge in the ABP. Similarly, the polynomial $[a, v_i]$ can be written as

$$[a, v_i] = \sum_{j=1}^{r'} [a, u_j] \cdot [u_j, v_i] \ .$$

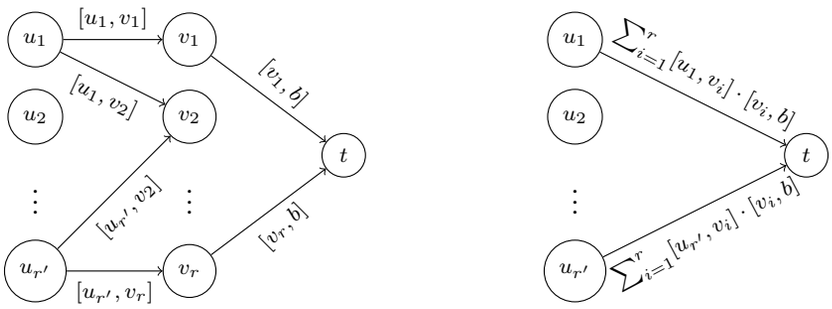


Figure 2.3.: Illustration of the removal the last layer of the ABP when all the edges to the sink vertex are constants, as in Claim 2.4.10. The figure only shows the last layers of the ABP. For every $i \in [r]$, the label $[v_i, b]$ is a field constant.

Combining the two expressions together, we get

$$[a, b] = \sum_{i=1}^{r} [v_i, b] \cdot \left(\sum_{j=1}^{r'} [u_j, v_i] \cdot [a, u_j] \right) ,$$

which on further rearrangement, gives us

$$[a, b] = \sum_{j=1}^{r'} \left(\sum_{i=1}^{r} [v_i, b][u_j, v_i] \right) \cdot [a, u_j] \,. \tag{2.4.11}$$

From the hypothesis of the claim, we know that for every $i \in [r]$, the edge label $[v_i, b]$ is a field constant, and the edge label $[u_j, v_i]$ is a polynomial of degree at most Δ. Thus, for every $j \in [r']$, the expression $(\sum_{i=1}^{r} [u_i, b][u_j, v_i])$ is a polynomial of degree at most Δ.

This gives us the following natural construction for the ABP $\mathcal{M}'$ from $\mathcal{M}$. We delete the vertices $v_1, v_2, \ldots, v_r$ in $\mathcal{M}$ (and hence, all edges incident to them), and for every $j \in \{1, 2, \ldots, r'\}$, we connect the vertex u_j with the end vertex b using an edge with label $(\sum_{i=1}^{r} [v_i, b][u_j, v_i])$ (see also Figure 2.3). The upper bound on the size and the number of layers of $\mathcal{M}'$ is immediate from the construction, and that it computes the same polynomial as $\mathcal{M}$ follows from Equation 2.4.11. □

We now prove a simple generalization of Lemma 2.4.8 for a multilayered ABP.

Lemma 2.4.12. *Let $\mathcal{A} = \sum_{i=1}^{m} \mathcal{A}_i$ be a multilayered ABP with d layers over a field $\mathbb{F}$ computing the polynomial F, such that each $\mathcal{A}_i$ is an ABP with d_i layers. Also, let $\ell_{i,j}$ be the number of vertices in the j-th layer of $\mathcal{A}_i$ ($\ell_{i,j} = 0$ if $\mathcal{A}_i$ has fewer than j layers), and $\ell = \min_{j \in (d/3, 2d/3)} \{\sum_{i=1}^{m} \ell_{i,j}\}$.*

Then, there is a multilayered ABP with at most $2d/3$ layers and size at most $|\mathcal{A}|$ that computes a polynomial of the form

$$F - \sum_{i=1}^{\ell} P_i \cdot Q_i + \delta\,,$$

where $\{P_1, \ldots, P_\ell, Q_1, \ldots, Q_\ell\}$ is a set of non-constant polynomials with constant term zero and $\delta \in \mathbb{F}$.

Proof. Let $j_0 \in (d/3, 2d/3)$ be the natural number which minimizes the quantity $\sum_{i=1}^{m} \ell_{i,j}$, and let $S \subseteq [m]$ be the set of all indices i such that $\mathcal{A}_i$ has at least j_0 layers. Let $\mathcal{A}' = \sum_{i \in S} \mathcal{A}_i$ and $\mathcal{A}'' = \sum_{i \notin S} \mathcal{A}_i$. Thus,

$$\mathcal{A} = \mathcal{A}' + \mathcal{A}''.$$

Here, $\mathcal{A}'' = \sum_{i \notin S} \mathcal{A}_i$ is a multilayered ABP with at most $2d/3$ layers. Moreover, $|\mathcal{A}| = |\mathcal{A}'| + |\mathcal{A}''|$.

The idea now is to apply Lemma 2.4.8 to every ABP in $\mathcal{A}'$. For every $i \in S$, we know that there exist some polynomials $P_{i,1}, \ldots, P_{i,\ell_{i,j_0}}, Q_{i,1}, \ldots, Q_{i,\ell_{i,j_0}}$ with constant terms zero and a constant δ_i, such that

$$F_i - \sum_{r=0}^{\ell_{i,j_0}} P_{i,r} Q_{i,r} + \delta_i$$

can be computed by a multilayered ABP. Let us denote this multilayered ABP by $\mathcal{B}_i$.

From Lemma 2.4.8, we know that $\mathcal{B}_i$ has at most $\max\{j_0, d_i - j_0 + 1\} \leq 2d/3$ layers and size at most $|\mathcal{A}_i|$. Taking a sum over all $i \in S$ and re-indexing the summands, we get that there exist polynomials $P_1, \ldots, P_\ell, Q_1, \ldots, Q_\ell$ with constant terms zero and a constant δ such that the polynomial

$$\sum_{i \in S} F_i - \sum_{r=0}^{\ell} P_r Q_r + \delta$$

is computable by a multilayered ABP $\mathcal{B} = \sum_{i \in S} \mathcal{B}_i$ with at most $2d/3$ layers and size at most $\sum_{i \in S} |\mathcal{A}_i| \leq |\mathcal{A}'|$. Now, by combining the multilayered ABPs $\mathcal{B}$ and $\mathcal{A}''$, we get that the polynomial

$$F - \sum_{r=0}^{\ell} P_r Q_r + \delta$$

is computable by a multilayered ABP of size at most $|\mathcal{A}|$ and at most $2d/3$ layers. □

We now use Lemma 2.4.12 to prove the following claim.

Claim 2.4.13. *Let $\mathcal{A}_k$ be a multilayered ABP with edge labels of degree at most Δ, $d_k \geq n/\Delta$ layers and size at most τ that computes a polynomial of the form*

$$\sum_{i=1}^{n} x_i^n + \sum_{j=1}^{r} A_j \cdot B_j + R.$$

where $A_1(\mathbf{x}), \ldots, A_r(\mathbf{x})$, $B_1(\mathbf{x}), \ldots, B_r(\mathbf{x})$ are polynomials such that for every j, $A_j(\mathbf{0}) = B_j(\mathbf{0}) = 0$ and $R(\mathbf{x})$ is a polynomial of degree at most $n-1$.

If $\tau \leq 0.001n^2/\Delta$, then there exists a multilayered ABP $\mathcal{A}_{k+1}$ with at most $2d_k/3$ layers and size at most τ which computes a polynomial of the form

$$\sum_{i=1}^{n} x_i^n + \sum_{j=1}^{r'} A'_j \cdot B'_j + R' \,,$$

such that $r' \leq r + 0.005\frac{n^2}{\Delta \cdot d_k}$, $A'_1(\mathbf{x}), \ldots, A'_{r'}(\mathbf{x})$, $B'_1(\mathbf{x}), \ldots, B'_{r'}(\mathbf{x})$ are polynomials such that for every i, $A'_i(\mathbf{0}) = B'_i(\mathbf{0}) = 0$ and $R'(\mathbf{x})$ is a polynomial of degree at most $n-1$.

Proof. Let $\mathcal{A}_k = \sum_{i=1}^{m} \mathcal{A}_{k,i}$, and for $j \in [d_k]$, let $\ell_{i,j}$ be the number of vertices in layer j of $\mathcal{A}_{k,i}$. Recall that if the number of layers in $\mathcal{A}_{k,i}$ is strictly less than j, then we set $\ell_{i,j} = 0$. Let ℓ be the total number of vertices in the middle layers of $\mathcal{A}_k$, defined as

$$\ell = \sum_{i=1}^{m} \left(\sum_{j \in (d_k/3, 2d_k/3)} \ell_{i,j} \right) .$$

Since $\ell \leq \tau \leq \frac{0.001n^2}{\Delta}$, by averaging, we know that there is a $j_0 \in (d_k/3, 2d_k/3)$, such that

$$\ell_{j_0} = \sum_{i=1}^{m} \ell_{i,j_0} \leq \frac{\ell}{d_k/3} \leq \frac{0.001n^2}{\Delta} \cdot \frac{1}{d_k/3} \leq 0.005 \frac{n^2}{\Delta \cdot d_k} \,.$$

This condition, together with Lemma 2.4.12, tells us that there is a multilayered ABP $\mathcal{A}'_{k+1}$ with at most $2d_k/3$ layers and size at most τ that computes a polynomial of the form

$$\sum_{i=1}^{n} x_i^n + \sum_{j=1}^{r} A_j \cdot B_j + R - \sum_{i=1}^{\ell_{j_0}} P_i \cdot Q_i + \delta \,,$$

where $P_1, \ldots, P_{\ell_{j_0}}, Q_1, \ldots, Q_{\ell_{j_0}}$ are a set of non-constant polynomials with constant term zero and $\delta \in \mathbb{F}$. Since $\ell_{j_0} \leq 0.005 \frac{n^2}{\Delta \cdot d_k}$, the claim follows. □

We are finally ready to prove our main theorem, which we restate once more.

Theorem 2.4.14. *Let $\mathbb{F}$ be a field and $n \in \mathbb{N}$ such that $\mathrm{char}(\mathbb{F}) \nmid n$. Then any algebraic branching program over $\mathbb{F}$ computing the polynomial $\sum_{i=1}^{n} x_i^n$ is of size at least $\Omega(n^2)$.*

When the ABP's edge labels are allowed to be polynomials of degree at most Δ, our lower bound is $\Omega(n^2/\Delta)$.

Proof. Let $\mathcal{A}$ be a multilayered ABP computing the polynomial $\sum_{i=1}^{n} x_i^n$ that has d_0 layers. As before we may assume, without loss of generality, that the underlying field $\mathbb{F}$ is algebraically closed.

Note that if d_0 is at most n/Δ, then the formal degree of $\mathcal{A}$ is at most $d_0\Delta \leq n$. Thus, by Theorem 2.4.5, we know that $|\mathcal{A}|$ is at least $\Omega(n^2/\Delta)$ and we are done. Also, if $d_0 > n^2/\Delta$, then again we have our lower bound since each layer of $\mathcal{A}$ must have at least one vertex. Thus, we can assume that $n/\Delta \leq d_0 \leq n^2/\Delta$.

The proof idea is to iteratively make changes to $\mathcal{A}$ till we get a multilayered ABP $\mathcal{A}'$ of formal degree at most n that computes a polynomial of the type

$$\sum_{i=1}^{n} x_i^n + \sum_{j=1}^{r} A_j \cdot B_j + R$$

where $r \leq n/10$ and $A_1(\mathbf{x}), \ldots, A_r(\mathbf{x}), B_1(\mathbf{x}), \ldots, B_r(\mathbf{x}), R(\mathbf{x})$ are polynomials such that for every i, $A_i(\mathbf{0}) = B_i(\mathbf{0}) = 0$ and R has degree at most $n-1$. Once we have this, we can invoke Theorem 2.4.5 and get the required lower bound.

We now explain how to obtain $\mathcal{A}'$ from $\mathcal{A}$ using Claim 2.4.13. Let us set $\mathcal{A}_0 = \mathcal{A}$. Then, $\mathcal{A}_0$ is a multilayered ABP with d_0 layers and size at most τ that computes the polynomial $\sum_{i=1}^{n} x_i^n$.

If $\tau \geq 0.001n^2/\Delta$, the statement of the theorem follows. Otherwise, we apply Claim 2.4.13 iteratively K times, as long as the number of layers is more than n/Δ, to eventually get a multilayered ABP $\mathcal{A}' = \mathcal{A}_K$ with $d' \leq n/\Delta$ layers. Let $d_0, \ldots, d_{K-1}, d_K$ denote the number of layers in each ABP in this sequence, so that $d_{K-1} > n/\Delta$, and $d_k \leq 2d_{k-1}/3$ for $k \in [K]$. $\mathcal{A}'$ is an ABP with at most n/Δ layers and size at most τ, which by induction, computes a polynomial of the form

$$\sum_{i=1}^{n} x_i^n + \sum_{j=1}^{r} A_j \cdot B_j + R\,,$$

where $A_1(\mathbf{x}), \ldots, A_r(\mathbf{x})$, $B_1(\mathbf{x}), \ldots, B_r(\mathbf{x})$ are polynomials such that for every i, $A_i(\mathbf{0}) = B_i(\mathbf{0}) = 0$ and $R(\mathbf{x})$ is a polynomial of degree at most $n-1$. Further, the number of error terms, r, is at most

$$\frac{0.005n^2}{\Delta}\left(\frac{1}{d_{K-1}} + \frac{1}{d_{K-2}} + \cdots + \frac{1}{d_0}\right).$$

Since $d_k \leq \frac{2}{3} \cdot d_{k-1}$, we have that $\frac{1}{d_{k-1}} \leq \frac{2}{3} \cdot \frac{1}{d_k}$ for all $k \in [K]$, so that

$$r \leq \frac{0.005n^2}{\Delta} \cdot \frac{1}{1-2/3} \cdot \frac{1}{d_{K-1}} \leq \frac{n}{10}$$

as $d_{K-1} \geq n/\Delta$.

At this point, since the formal degree is at most n, using Theorem 2.4.5 we get

$$\tau \geq |\mathcal{A}'| \geq \frac{(n/2-r)n}{2\Delta} = \Omega\left(\frac{n^2}{\Delta}\right). \qquad \square$$

2.5 Unlayered Algebraic Branching Programs

In this section, we prove Theorem 2.2.2. We begin with the following definition.

Definition 2.5.1. *Let $\mathcal{A}$ be an unlayered ABP over $\mathbb{F}$. Let s and t denote the start and end vertices of $\mathcal{A}$, respectively, and let $v \neq s, t$ be a vertex in $\mathcal{A}$. Denote by $\alpha \in \mathbb{F}$ the constant term of $[s, v]$ and by $\beta \in \mathbb{F}$ the constant term of $[v, t]$.*

The cut *of $\mathcal{A}$ with respect to v, denoted $\mathsf{cut}(\mathcal{A}, v)$, is the unlayered ABP obtained from $\mathcal{A}$ using the following sequence of operations:*

1. *Duplicate the vertex v (along with its incoming and outgoing edges). Let v_1, v_2 denote the two copies of v.*
2. *Erase all outgoing edges of v_1, and connect v_1 to t by a new edge labelled β.*
3. *Erase all incoming edges of v_2, and connect s to v_2 by a new edge labelled α.* $\diamondsuit$

We now prove some basic properties of the construction in Definition 2.5.1.

Claim 2.5.2. *Let $\mathcal{A}$ be an unlayered ABP over $\mathbb{F}$ computing a polynomial F, and let v be a vertex in $\mathcal{A}$. Denote $\mathcal{A}' = \mathsf{cut}(\mathcal{A}, v)$. Denote by d the depth of $\mathcal{A}$ and by d_v the depth of v in $\mathcal{A}$. Then the following properties hold:*

1. *$\mathcal{A}'$ has 1 more vertex and 2 more edges than $\mathcal{A}$.*

2. *The depth of $\mathcal{A}'$ is at most*

$$\max\left\{\mathsf{depth}(\mathcal{A}\setminus\{v\}), d_v+1, d-d_v+1\right\},$$

where $\mathcal{A}\setminus\{v\}$ is the ABP obtained from $\mathcal{A}$ by erasing v and all of its adjacent edges.

3. *$\mathcal{A}'$ computes a polynomial of the form $F - P\cdot Q - \delta$ where P and Q have no constant term, and $\delta\in\mathbb{F}$.*

Proof. The first property is immediate from the construction. The second property follows from the following reasoning: each path in $\mathcal{A}'$ is of exactly one of the following types: (a) misses both v_1 and v_2, (b) passes through v_1, or (c) passes through v_2. In case (a), the path also appears in the graph of $\mathcal{A}\setminus\{v\}$. In case (b), the only edge going out of v_1 is to t, and all other edges in the path appear in $\mathcal{A}$, hence the length is at most d_v+1. In case (c), the only edge entering v_2 is from s, hence similarly the path is of length at most $d-d_v+1$.

It remains to show the last property. Let $P'=[s,v]$ and $Q'=[v,t]$ (as computed in $\mathcal{A}$). Denote $P'=P+\alpha$ where P has no constant term and $\alpha\in\mathbb{F}$ and similarly $Q'=Q+\beta$. One may write $F=P'\cdot Q'+R=(P+\alpha)(Q+\beta)+R$ where R is the sum over all paths in $\mathcal{A}$ which do not pass through v. In $\mathcal{A}'$, we have that $[s,v_1]=P'$ and $[v_2,t]=Q'$, and thus $\mathcal{A}'$ computes the polynomial

$$R+\alpha\cdot Q'+P'\cdot\beta = F-P\cdot Q+\alpha\beta. \qquad \square$$

Our goal is to perform cuts on a strategically chosen set of vertices. In order to select them, will use the following well known lemma of [Val77], simplifying and improving an earlier result of Erdős, Graham and Szemerédi [EGS75]. For completeness, we also sketch a short proof.

Lemma 2.5.3 ([Val77])**.** *Let G be a directed acyclic graph with m edges and depth $d\geq\sqrt{n}$. Then, there exists a set E' of at most $4m/\log n$ edges such that removing E' from G results in a graph of depth at most $d/2$.*

Proof. Let $d'\geq d\geq\sqrt{n}$ be a smallest power of 2 larger than d, so that $d'\leq 2d$. Let $k=\log d'$. A *valid labeling* of a directed graph $G=(V,E)$ is a function $f:V\to\{0,\ldots,N-1\}$ such that whenever (u,v) is an edge, $f(u)<f(v)$.

Clearly if G had depth d then there is a valid labeling with image $\{0,\ldots,N-1\} = \{0,\ldots,d-1\}$ by labeling each vertex by its depth. Conversely, if there is a valid labeling with image $\{0,\ldots,N-1\}$ then $\mathsf{depth}(G) \leq N$.

Let f be a valid labeling of G with image $\{0,\ldots,d'-1\}$ and for $i \in [k]$ let E_i be the set of edges such that the most significant bit in which the binary encoding of the labels of their endpoints differ is i. If E_i is removed, we can obtain a valid relabelling of the graph with image $\{0,\ldots,d'/2-1\}$ by removing the i-th bit from all labels.

The two smallest sets among the E_i-s have size at most $2m/k$, which is at most $4m/\log n$ (since $k = \log d' \geq \log n/2$). Thus removing them gives a valid labeling with image $\{0,\ldots,d'/4-1\}$ and therefore a graph with depth at most $d'/4 \leq d/2$. □

We need a slight variation of this lemma, in which we do not pick edges whose endpoints have too small or too large a depth in the graph.

Lemma 2.5.4. *Let G be a directed acyclic graph with m edges and depth $d \geq \sqrt{n}$. Then, there exist a set U of vertices, of size at most $4m/\log n$, such that for every $v \in U$ we have that $d/9 \leq \mathsf{depth}(v) \leq 8d/9$, and removing U (and the edges touching those vertices) results in a graph of depth at most $3d/4$.*

Proof. Let E denote the set of edges of G and $E' \subseteq E$ be the set of edges guaranteed by Lemma 2.5.3. Let $E_1 \subseteq E'$ be the edges in E' whose heads have depth at most $d/9$, and E_2 be the edges in E' whose heads have depth at least $8d/9$. Let $E'' = E' \setminus (E_1 \cup E_2)$ (see also Figure 2.4). Clearly, $|E''| \leq |E'| \leq 4m/\log n$. Let U be the set of heads of edges in E''.

Consider now any path in the graph obtained from G by removing U (and hence in particular E''). Given such a path, let e_1 be the last edge from E_1 in the path which appears before all edges from E_2 (if there exists such an edge), and let e_2 the first edge from E_2 (if any) in the path. We partition the path into three (possibly empty) parts: the first part is all the edges which appear until e_1 (including e_1); the second part is all the edges after e_1 and before e_2; the last part consists of all the edges which appear after e_2 (including e_2). Because the head of e_1 is a vertex of depth at most $d/9$, the first part can contribute at most $d/9$ edges. The second part includes only edges from $E \setminus E'$, and thus its length is at most $d/2$. The last part again has depth at most $d/9+1$, as any path leaving a vertex of depth at least $8d/9$ can have at most that many edges (here we add 1 to account for the edge e_2 itself,

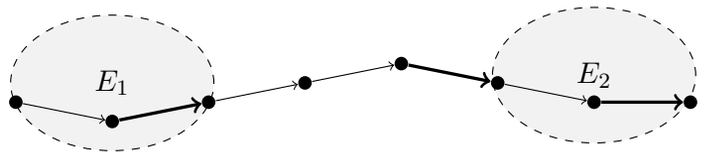


Figure 2.4.: The types of edges in Lemma 2.5.4. Thick edges were selected to E' by Lemma 2.5.3. The sets E_1 contains the edges in E' of low depth and E_2 the edges in E' of large depth. E'' is constructed by removing E_1 and E_2 edges from E'.

since the assumption is on the depth of the head of e_2). Thus, the total length of the path is at most

$$d/9 + d/2 + d/9 + 1 \leq 3d/4\,. \qquad \square$$

The set of vertices given by the lemma above will be the vertices according to which we will cut the ABP. We describe it in the following lemma, and prove some properties of this operation.

Lemma 2.5.5. *Let $\mathcal{A}$ be an unlayered ABP over a field $\mathbb{F}$ of depth $d \geq \sqrt{n}$ computing a polynomial F. Let τ be the number of vertices and m be the number of edges in $\mathcal{A}$. Then there exist an unlayered ABP $\mathcal{A}'$, with at most $\tau + 4m/\log n$ vertices, at most $m + 8m/\log n$ edges, and depth at most $9d/10$, computing a polynomial of the form $F - \sum_{i=1}^{r} P_i Q_i - \delta$ where $\delta \in \mathbb{F}$ is a field constant, the P_i, Q_i's have no constant term, and $r \leq 4m/\log n$.*

Proof. Let G be the underlying graph of the unlayered ABP $\mathcal{A}$. Let $U = \{u_1, \ldots, u_r\}$ be the set of vertices guaranteed by Lemma 2.5.4, such that $r \leq 4m/\log n$. We perform the following sequence of cuts on $\mathcal{A}$. Set $\mathcal{A}_0 := \mathcal{A}$ and for $i \in [r]$, $\mathcal{A}_i = \mathsf{cut}(\mathcal{A}_{i-1}, u_i)$. Finally $\mathcal{A}' = \mathcal{A}_r$.

The statements of the lemma now follow from the properties of cuts as proved in Claim 2.5.2. The bound on the number of vertices and edges in $\mathcal{A}'$ is immediate. The claim on the polynomial computed by $\mathcal{A}'$ follows by induction on i.

Finally, by induction on i, we have that the depth of $\mathcal{A}'$ is at most the maximum among $\mathsf{depth}(u_1)+1, \ldots, \mathsf{depth}(u_r)+1$, $d - \mathsf{depth}(u_1)+1, \ldots, d - \mathsf{depth}(u_r)+1$ and $\mathsf{depth}(\mathcal{A} \setminus U)$, where $\mathcal{A} \setminus U$ is the ABP obtained by removing all vertices in U.

By the choice of U as in Lemma 2.5.4, for every $i \in [r]$ we have that $d/9 \leq \mathsf{depth}(u_i) \leq 8d/9$, and $\mathsf{depth}(\mathcal{A} \setminus U) \leq 3d/4$, which implies the required upper bound on the depth of $\mathcal{A}'$ (assuming n, and hence d, are large enough). $\square$

Repeated applications of Lemma 2.5.5 give the following statement.

Corollary 2.5.6. *Let $\mathcal{A}$ be an unlayered ABP over a field $\mathbb{F}$, with edge labels of degree at most $\Delta = n^{o(1)}$, computing an n-variate polynomial F. Further suppose $\mathcal{A}$ has depth at least $\sqrt{n}$, and that the number of edges in $\mathcal{A}$ is at most*

$$n \log n/(1000(\log\log n + \log \Delta)) \,.$$

Let τ denote the number of vertices in $\mathcal{A}$.

Then, there exists an unlayered ABP $\mathcal{A}'$, whose depth is at most n/Δ, which computes a polynomial of the form

$$F - \sum_{i=1}^{r} P_i Q_i - \delta \,,$$

such that P_i, Q_i are all polynomials without a constant term, $\delta \in \mathbb{F}$ is a field constant, and $r \leq n/10$. The number of vertices in $\mathcal{A}'$ is at most $\tau + n/10$.

Proof. Observe that the depth of $\mathcal{A}$ is at most $d := n \log n$. As long as the depth is at least $\sqrt{n}$, apply Lemma 2.5.5 repeatedly at most $k := 7(\log\log n + \log \Delta)$ times, to obtain an unlayered ABP of depth at most $(0.9)^k \cdot d \leq n/\Delta$.

The upper bound on the number of summands P_iQ_i and the number of vertices after each application is given as a function of the number of edges, which increases in the process. Hence, we first provide a crude estimate on the number of edges at each step. For $i \in [k]$, let $\mathcal{A}_i$ denote the unlayered ABP obtained after the i-th application of Lemma 2.5.5, and let m_i be the number of edges in $\mathcal{A}_i$.

We claim that by induction on i, $m_i \leq m_0 \cdot (1 + 8/\log n)^i$. This is true for $i = 0$ by definition. For $i \geq 1$, since we maintain the invariant that the depth is at least $\sqrt{n}$, it follows from Lemma 2.5.5 that

$$\begin{aligned} m_i \leq m_{i-1} + 8m_{i-1}/\log n &= m_{i-1}(1 + 8/\log n) \\ &\leq m_0(1 + 8/\log n)^{i-1} \cdot (1 + 8/\log n) \end{aligned}$$

where the last inequality uses the induction hypothesis.

Thus, the final unlayered ABP has at most

$$\begin{aligned} m_k \leq m_0(1 + 8/\log n)^k &\leq 2m_0 \\ &= n \log n/(500(\log\log n + \log \Delta)) =: M \end{aligned}$$

assuming n is large enough (recall that by assumption we have that $\log \Delta = o(\log n)$, so that $\lim_{n\to\infty}(1+8/\log n)^{o(\log n)} = 1$). It is convenient to now use M as a uniform upper bound on the number of edges in all stages of this process, so that each step adds at most $4M/\log n$ summands and vertices. It now follows that r is at most

$$\frac{4kM}{\log n} \leq \frac{7(\log\log n + \log\Delta)\cdot 4n}{500(\log\log n + \log\Delta)} \leq n/10,$$

and similarly the total number of vertices added throughout the process is at most $n/10$. □

The lower bound given in Theorem 2.2.2 now follows by a simple win-win argument. For convenience, we restate the theorem.

Corollary 2.5.7. *Let $\mathcal{A}$ be an unlayered ABP over a field $\mathbb{F}$, with edge labels of degree at most $\Delta = n^{o(1)}$, computing $\sum_{i=1}^{n} x_i^n$. Then $\mathcal{A}$ has at least $\Omega(n\log n/(\log\log n + \log\Delta)$ edges.*

Proof. Let τ denote the number of vertices in $\mathcal{A}$. If the number of edges is at least $n\log n/(1000(\log\log n + \log\Delta))$, then we already have our lower bound. Else, the number of edges is at most $n\log n/(1000(\log\log n+\log\Delta))$. Now, by Corollary 2.5.6, there exists an unlayered ABP $\mathcal{A}'$, with $\tau + n/10$ vertices and depth at most n/Δ, computing $\sum_{i=1}^{n} x_i^n - \sum_{j=1}^{r} P_iQ_i - \delta$, such that P_j, Q_j have no constant term, $r \leq n/10$, and $\delta \in \mathbb{F}$.

It thus follows that $\mathcal{A}'$ has formal degree at most n. By Theorem 2.4.5, it has $\Omega(n^2/\Delta)$ vertices, thus $\tau = \Omega(n^2/\Delta)$, so that the number of edges is also $\Omega(n^2/\Delta)$. □

A natural open question here is to prove an improved lower bound for unlayered algebraic branching programs. In particular, in the absence of an obvious non-trivial upper bound, it seems reasonable to conjecture that any unlayered ABP computing the polynomial $\sum_{i=1}^{n} x_i^n$ has size at least $\Omega(n^{2-o(1)})$.

3 Lower Bounds Against General Algebraic Formulas

In this chapter, we study the model of algebraic formulas. Recall that they are algebraic circuits where the underlying graph is a tree. For polynomials like $\sum_{i=1}^{n} x_i^n$, a quadratic lower bound on the formula size is immediate. This is because the degree of the polynomial in every variable is n, and hence there must be at least n leaves which are labelled by x_i for every i. Therefore, for a formula lower bound to be non-trivial it should hold for a polynomial with bounded individual degrees. We discuss this further while discussing previous work.

3.1 Our Results

Our main result in this chapter is an $\Omega(n^2)$ lower bound against algebraic formulas for a multilinear polynomial.

Theorem 3.1.1. *Let $n \in \mathbb{N}$ and let $\mathbb{F}$ be a field of characteristic greater than $0.1n$. Then any algebraic formula over $\mathbb{F}$ computing the elementary symmetric polynomial*

$$\mathrm{ESYM}_{,(n,0.1n)}(\mathbf{x}) = \sum_{S \subseteq [n], |S| = 0.1n} \prod_{j \in S} x_j \,,$$

is of size at least $\Omega(n^2)$.

As we will describe while discussing the proof overview, even though formulas can be simulated by ABPs with little overhead, this theorem is not an immediate corollary of Theorem 2.2.1 and requires some work. Before we get into those details, let us look at some prior work on formula lower bounds.

Previous Work

As mentioned earlier, note that any algebraic formula computing the polynomial $\sum_{i=1}^{n} x_i^n$ has at least n^2 leaves, as since the degree of the polynomials in each of

the variables x_i is n, there must be at least n leaves in the formula which are labelled by x_i. Thus, the number of leaves is at least $n \times n = n^2$. Therefore, for lower bounds for algebraic formulas to be interesting and non-trivial, they must be asymptotically larger than the sum of the individual degrees of the variables. One particularly interesting regime of parameters is when the target hard polynomial has constant individual degree (which can be taken to be 1, essentially without loss of generality).

The first non-trivial lower bound for algebraic formulas was due to [Kal85] who proved a lower bound of $\Omega(n^{3/2})$ for an n-variate multilinear polynomial (in fact for the $\sqrt{n} \times \sqrt{n}$ determinant). Kalorkoti's proof is essentially an algebraic version of the one by [Nec66] which showed a lower bound of $\Omega(n^2/\log n)$ on the size of Boolean formula in the full binary basis (gates are allowed to be all Boolean functions on 2 variables). Using very similar arguments, it is possible to show an $\Omega(n^2/\log n)$ lower bound for a different multilinear polynomial (the construction we describe here is based on a non-multilinear construction given in [SY10]): let $\left\{x_i^{(j)} : i \in [\log n], j \in [n/\log n]\right\}$ and $\{y_k : k \in [n]\}$ be two sets of n variables each, and fix a bijection $\iota : 2^{[\log n]} \to [n]$. The hard multilinear polynomial will be

$$\sum_{j=1}^{n/\log n} \left(\sum_{S \subseteq [\log n]} y_{\iota(S)} \cdot \prod_{i \in S} x_i^{(j)} \right).$$

It turns out that in general, this technique of Nechipurok and Kalorkoti *cannot* prove a lower bound which is asymptotically better than $(n^2/\log n)$ (see, for example, [Juk12, Remark 6.18]).

In Theorem 3.1.1, we adapt the ideas used in Theorem 2.2.1 to prove an $\Omega(n^2)$ lower bound for an explicit n variate multilinear polynomial. Thus, while the quantitative improvement over the previous lower bound is by a mere $\log n$ factor, it is interesting to note that Theorem 3.1.1 gives an asymptotically better lower bound than the best bound that can be proved using the classical, well known techniques of Nechiporuk and Kalorkoti.

The n-variate elementary symmetric polynomial of degree $0.1n$ which, due to a well known observation of Ben-Or, is also known to be computable by a formula (in fact, a depth-3 formula) of size $O(n^2)$ over all large enough fields. For a large regime of parameters, this construction was shown to be tight for depth-3 formulas (even depth-3 circuits) by [SW01]. Theorem 3.1.1 shows that elementary symmetric polynomials do not have formulas of size $o(n^2)$ regardless of their depth. Prior to this work, the only super linear lower bound on the general formula complexity of

elementary symmetric polynomials we are aware of is the $\Omega(n \log n)$ lower bound of [BS83], which as we mentioned earlier is in fact a lower bound on the stronger notion of *circuit* complexity. As far as we understand, Kalorkoti's proof directly does not seem to give any better lower bounds on the formula complexity of these polynomials.

Another mildly interesting point is that there is also an *upper bound* of $O(n \log n)$[1] on the circuit complexity of the elementary symmetric polynomials due to [BS83]. Hence, our lower bound implies a super-linear separation between the formula complexity and the circuit complexity of an explicit multilinear polynomial family.

Proof Overview

The proof of Theorem 3.1.1 is similar to that of Theorem 2.2.1 but needs a few more ideas. A natural attempt at recovering formula lower bounds from ABP lower bounds is to convert the formula to an ABP and then invoke the lower bound in Theorem 2.2.1. However, while it is easy to see that a formula can be transformed into an ABP with essentially no blow up in size, the ABP obtained at the end of this transformation is not necessarily layered. Therefore, we cannot directly invoke Theorem 2.2.1. We can still invoke Theorem 2.2.2, but the lower bound thus obtained is barely super-linear and, in particular, weaker than the known lower bounds [Kal85; SY10].

For the proof of Theorem 3.1.1, we apply the ideas in the proof of Theorem 2.2.1 in a non black-box manner. The proof is again in two steps, where in the first step, we show a lower bound of $\Omega(n^2)$ for polynomials of the form $\mathrm{ESYM}_{,(n,0.1n)}(\mathbf{x}) + \varepsilon(\mathbf{x})$ for formulas of formal degree at most $0.1n$. Here, $\varepsilon(\mathbf{x})$ should again be thought of as an error term as in the outline of the proof of Theorem 2.2.1.

For formulas of formal degree greater than $0.1n$, we describe a simple structural procedure to reduce the formal degree of a formula, while maintaining its size. The cost of the procedure is that the complexity of error terms increases in the process. We argue that starting from a formula of size at most n^2 (since otherwise, we are already done) computing $\mathrm{ESYM}_{,(n,0.1n)}(\mathbf{x})$, we can repeatedly apply this degree reduction procedure to obtain a formula of formal degree at most $0.1n$ which computes the polynomial $\mathrm{ESYM}_{,(n,0.1n)}(\mathbf{x}) + \varepsilon(\mathbf{x})$, where the error terms is not too

[1] The key step in the proof refers to a paper that is written in German and we were unable to reconstruct it. Ramprasad, however, told us about an $O(n \log^2 n)$ upper bound which can also be found on cs.stackexchange.

complex. Combining this with the robust lower bound for formulas of formal degree at most $0.1n$ completes the proof of Theorem 3.1.1.

3.2 Preliminaries

Similar to the previous chapter, we need the notion of affine algebraic varieties and we refer the reader to section 2.3 for an overview of its definition and some of its properties. As before, all logarithms are base 2.

Properties of Elementary Symmetric Polynomials

We begin by defining the family of elementary symmetric polynomials formally.

$$\mathrm{ESYM}_{n,d}(\mathbf{x}) = \sum_{S\subseteq[n],|S|=d} \prod_{j\in S} x_j$$

We now discuss some properties of the polynomial family. These seem to be well known and are not original to this work.

Lemma 3.2.1 ([MZ17; LMP19])**.** *Let $n, d \in \mathbb{N}$ be natural numbers with $2 \leq d \leq n$. Then, over any field of characteristic at least $d+1$, the dimension of the variety V defined as*

$$V = \mathbb{V}\left(\left\{\partial_{x_i}\mathrm{ESYM}_{,(n,d)} \ : \ i \in [n]\right\}\right)$$

is at most $d-2$. Here, $\partial_{x_i}\mathrm{ESYM}_{n,d}(\mathbf{x})$ is the partial derivative of $\mathrm{ESYM}_{,(n,d)}$ with respect to the variable x_i.

We quickly sketch the outline of a proof of this lemma from [MZ17] for completeness. The following claim immediately implies the lemma.

Claim 3.2.2. *Let $\mathbf{a} \in \mathbb{F}^n$ be a point in the variety V as defined in Lemma 3.2.1. Then, at least $n-(d-2)$ coordinates of $\mathbf{a}$ are equal to 0.*

To see that the lemma follows from the claim, observe that Claim 3.2.2 implies that the variety V is a subset of set of $\cup_{S\subseteq n,|S|=d-2} V_S$, where $V_S(\subseteq \mathbb{F}^n)$ is the set of points in $\mathbb{F}^n$ where the coordinates in S are all set to zero, and the other coordinates

take all possible values in $\mathbb{F}$. Thus, each V_S is a variety of dimension $d-2$, and V is contained in a union of such varieties. Therefore, the dimension of V cannot exceed $d-2$. We now sketch the proof of the claim. For that we need the following folklore fact, often attributed to Euler.

Fact 3.2.3 (Differentiation of Homogeneous Polynomials)**.** *Suppose $A(x_1,\ldots,x_k)$ is a homogeneous polynomial of degree t. Then $\sum_{i=1}^{k} x_i \cdot \frac{\partial A}{\partial x_i} = t \cdot A(x_1,\ldots,x_k)$.*

Proof of Claim 3.2.2. The proof is via induction on the degree of the elementary symmetric polynomials being considered. Note that we only consider the cases when the degree, d, satisfies $2 \le d \le n$.

For the base case, we show that the claim is true whenever $2 = d \le n$. In this case, the partial derivative $\partial_{x_i}\mathrm{ESYM}_{n,2}(\mathbf{x}) = \sum_{j\neq i} x_j$. Now for any $\mathbf{a} \in \mathbb{F}^n$, all the linear forms $\left\{\sum_{j\neq i} a_j\right\}_{i\in[n]}$ vanish exactly when each a_j is zero. This implies the claim.

For the induction step, let us fix $d > 2$ and assume that the claim holds for $\mathrm{ESYM}_{n,r}(\mathbf{x})$ for all $r \in \mathbb{N}$ with $2 \le r \le d-1$ and $n \ge r$. We then prove the claim for $\mathrm{ESYM}_{n,d}(\mathbf{x})$ for every $n \ge d$.

For any $n \ge d$, from the definition of elementary symmetric polynomials, observe that

$$\partial_{x_i}\mathrm{ESYM}_{n,d}(\mathbf{x}) = \mathrm{ESYM}_{n,d-1}(\mathbf{x}) - x_i \cdot \partial_{x_i}\mathrm{ESYM}_{n,d-1}(\mathbf{x})\,. \tag{3.2.4}$$

If we add up the equations above for each $i \in [n]$, we get

$$\sum_{i\in[n]} \partial_{x_i}\mathrm{ESYM}_{n,d}(\mathbf{x}) = n \cdot \mathrm{ESYM}_{n,d-1}(\mathbf{x}) - \sum_{i\in[n]} x_i \cdot \partial_{x_i}\mathrm{ESYM}_{n,d-1}(\mathbf{x})\,.$$

From Fact 3.2.3, this can be simplified to

$$\sum_{i\in[n]} \partial_{x_i}\mathrm{ESYM}_{n,d}(\mathbf{x}) = n \cdot \mathrm{ESYM}_{n,d-1}(\mathbf{x}) - (d-1) \cdot \mathrm{ESYM}_{n,d-1}(\mathbf{x})\,.$$

Thus, it follows that for every $\mathbf{a} \in \mathbb{F}^n$ where all the first order partial derivatives of $\mathrm{ESYM}_{n,d}(\mathbf{x})$ vanish, it must be the case that $\mathrm{ESYM}_{n,d-1}(\mathbf{x})$ is also zero at $\mathbf{a}$. Moreover, from Equation 3.2.4, this implies that each of the polynomials in the set

$$\{x_i \cdot \partial_{x_i}\mathrm{ESYM}_{n,d-1}(\mathbf{x}) \;:\; i \in [n]\}$$

must also vanish at $\mathbf{a}$. Now if k of the coordinates of $\mathbf{a}$ are equal to 0, and S is the subset of non-zero coordinates of $\mathbf{a}$, it must be the case that the polynomials

$$\{\partial_{x_i}\mathrm{ESYM}_{n-k,d-1}(\mathbf{x}) \ : \ i \in S\}$$

vanish at the point $\mathbf{a}_S \in \mathbb{F}^{n-k}$, where $\mathbf{a}_S$ is the $n-k$ dimensional vector obtained by projecting $\mathbf{a}$ to the set S of its coordinates.

By the induction hypothesis, we know that for every $r \leq d-1$ and $m \geq r$, any common zero of all the first order partial derivatives of $\mathrm{ESYM}_{m,r}(\mathbf{x})$ must be zero in at least $m-(r-2)$ coordinates. But $\mathbf{a}_S$ is a common zero with support $n-k$ of all the first order partial derivatives of $\mathrm{ESYM}_{n-k,d-1}(\mathbf{x})$.

Thus, if $n-k \geq d-1 (\implies k \leq (n-d+1))$, then by the induction hypothesis,

$$\text{no. of zero co-ordinates in } \mathbf{a}_S = 0 \geq n-k-(d-3) \implies k \geq n-d+3$$

which would lead to a contradiction. The only other case then, is that

$$d-1 > n-k \implies k \geq n-(d-2).$$

Thus, $\mathbf{a}$ is zero on at least $n-(d-2)$ of its coordinates. □

We will also need the following strengthening of a result by Limaye, Mittal and Pareek [LMP19] in our proof.

Lemma 3.2.5 ([MZ17; LMP19])**.** *Let $n, d \in \mathbb{N}$ be natural numbers with $2 \leq d \leq n$ and let $\mathbb{F}$ be a field of characteristic greater than d. Let $R_1, R_2, \ldots, R_n \in \mathbb{F}[\mathbf{x}]$ be polynomials of degree at most $d-2$. Then, the dimension of the variety V defined as*

$$V = \mathbb{V}\left(\left\{\partial_{x_i}\mathrm{ESYM}_{,(n,d)} - R_i \ : \ i \in [n]\right\}\right)$$

is at most $d-2$.

For the sake of completeness, we provide a proof for a slightly more general statement. In this regard, we first introduce a couple of notations.

Notation 3.2.6. *For a polynomial f, let $\mathrm{Hom}_d(f)$ denote the d-th homogeneous component of f. Also, suppose we fix an ordering $<$ on the monomials in $\mathbb{F}[\mathbf{x}]$ that is graded*

according to degree. Then for any ideal $I \subseteq \mathbb{F}[\mathbf{x}]$, we denote by $\mathsf{LM}(I)$ the leading monomial ideal of I. That is,

$$\mathsf{LM}(I) = \{m \ : \ m \text{ is a leading monomial (w.r.t. } <) \text{ for some polynomial in } I\} .\Diamond$$

Lemma 3.2.7. *Let $\{f_1, \ldots, f_k\}$ be a set of homogeneous polynomials such that $\deg(f_i) = d_i$. Let $R_1, \ldots, R_k$ be polynomials such that $\deg(R_i) < d_i$, and let $I = \langle f_1, \ldots, f_k \rangle$ and $I' = \langle f_1 + R_1, \ldots . f_k + R_k \rangle$. Then,*

$$\dim(\mathbb{V}(I')) \leq \dim(\mathbb{V}(I))$$

Proof. The proof follows from the following claim.

Claim 3.2.8. *If $\{f_1, \ldots, f_k\}$ are homogeneous polynomials and $\deg(f_i) = d_i$, and let I, I' as in Lemma 3.2.7. Then $\mathsf{LM}(I) \subseteq \mathsf{LM}(I')$.*

Indeed, by the statement of the claim,

$$\begin{aligned} \mathsf{LM}(I) \subseteq \mathsf{LM}(I') \implies \mathbb{V}(\mathsf{LM}(I')) \subseteq \mathbb{V}(\mathsf{LM}(I)) \\ \text{(see, for example, [CLO07, Section 4.2])} \\ \implies \dim(\mathbb{V}(\mathsf{LM}(I'))) \leq \dim(\mathbb{V}(\mathsf{LM}(I))) \implies \dim(\mathbb{V}(I')) \leq \dim(\mathbb{V}(I)). \end{aligned}$$

The last implication follows from the fact that $\dim(\mathbb{V}(\mathsf{LM}(J))) = \dim(\mathbb{V}(J))$ for any ideal J (see, for example, [CLO07, Section 9.3]). □

Thus, to finish the proof of Lemma 3.2.7, we need to prove Claim 3.2.8.

Proof of Claim 3.2.8. Let $m \in \mathsf{LM}(I)$ be the leading monomial in $\sum_{i=1}^{k} g_i \cdot f_i$, and let $d = \deg(m)$. Since f_i is homogeneous of degree d_i, m continues to be the leading monomial in $\sum_{i=1}^{k} \mathrm{Hom}_{d-d_i}(g_i) \cdot f_i$, and hence in $\sum_{i=1}^{k} \mathrm{Hom}_{d-d_i}(g_i) \cdot (f_i + R_i)$. This shows that $m \in \mathsf{LM}(I')$. □

Using Lemma 3.2.1 and Lemma 3.2.7, it is easy to see that Lemma 3.2.5 holds.

3.3 A Lower Bound Against Algebraic Formulas

Theorem 3.3.1. *Any algebraic formula computing the polynomial* $\mathrm{ESYM}_{,(n,0.1n)}(\mathbf{x})$ *over a field of characteristic greater than* $0.1n$ *has size at least* $\Omega\left(n^2\right)$.

Recall that the polynomial we prove the lower bound for is the elementary symmetric polynomial on n-variables of degree $0.1n$, which is defined as follows.

$$\mathrm{ESYM}_{,(n,0.1n)}(\mathbf{x}) = \sum_{S\subseteq[n],|S|=0.1n} \prod_{j\in S} x_j$$

A few properties of the elementary symmetric polynomials are needed for our proof, and we first discuss them. These seem to be well known and are not original to this work.

A Degree Reduction Procedure

We will now prove a statement similar to Lemma 2.4.12 for the case of formulas. We start with a few preliminary remarks. It is common to define the size of the formula as the number of leaves (which is, up to a factor of 2, the total number of vertices in the formula). For us it is a bit more convenient to define the size of the formulas as the number of leaves labelled by *variables* (rather than field constants), and prove a lower bound on this measure.

The formal degree of a vertex in the formula is defined by induction in the natural way. If v is a leaf labelled by variable, its formal degree is 1. If v is a leaf labelled by a field constant, its formal degree is 0. The formal degree of a sum gate is the maximum between the formal degree of its children, and the formal degree of a product gate is the sum of the formal degrees of its children.

The formal degree of the formula is the formal degree of its output gate.

We start by a simple observation which can be proved by induction on the structure of the formula.

Observation 3.3.2. *The size of a formula is at least as large as its formal degree.*

To avoid cumbersome notation, we identify a formula Φ with the polynomial it computes. For a vertex v in the formula Φ, we denote by Φ_v subformula rooted at v (and similarly identify Φ_v with the polynomial it computes).

Lemma 3.3.3. *Let $\Phi \in \mathbb{F}[\mathbf{x}]$ be a formula of formal degree d. Then, for every $t \in \mathbb{N}$ such that $2t \leq d$, there is a vertex v in Φ of formal degree at least t and at most $2t-1$.*

Moreover, there are polynomials $h, f \in \mathbb{F}[\mathbf{x}]$ such that $\Phi \equiv h \cdot \Phi_v + f$ and for every $\gamma \in \mathbb{F}$, the polynomial $\gamma h + f \in \mathbb{F}[\mathbf{x}]$ can be computed by a formula of size at most $|\Phi| - |\Phi_v|$.

Proof. From the definition of formal degree, recall that for every $+$ gate of formal degree k, at least one of its children also has formal degree k, and for every $\times$ gate of formal degree k, at least one of its children has formal degree at least $\lceil k/2 \rceil$. To get our hands on the vertex v of formal degree in the interval $[t, 2t-1]$ in the formula Φ, we start from the root of Φ and traverse down, such that in every step if the degree of the current vertex w is at least $2t$, then we traverse down to its child with the larger degree (breaking ties arbitrarily). We stop the first time we reach a vertex v of formal degree at most $2t-1$. The claim is that such a vertex must have formal degree at least t.

To see the lower bound of t on the degree of v, recall that the formal degree of Φ is at least $2t$ and by definition, v is the first vertex on the unique path from v to the root of Φ with formal degree at most $2t-1$. Thus, the parent of v must have formal degree at least $2t$. We also know that at any stage of this walk down from the root, we go from a vertex to its child with the higher formal degree. Therefore, the formal degree of v is at least half of the degree of its parent, i.e., at least $\frac{1}{2} \cdot 2t = t$.

Consider now that formula obtained from Φ by replacing all the subtree rooted at v with a leaf labelled by a new variable y. This new formula Φ' computes a polynomial of the form $h \cdot y + f$, and by replacing y with Φ_v we recover the formula Φ, thus $\Phi \equiv h \cdot \Phi_v + f$.

The size of Φ' is $|\Phi| - |\Phi_v| + 1$. For every fixed $\gamma \in \mathbb{F}$ we can actually compute $\gamma h + f$ by replacing the leaf labelled y in Φ' by the constant γ (recall that we do not count leaves labelled by constant in or definition of size). □

We now use Lemma 3.3.3 inductively to get the following decomposition.

Lemma 3.3.4. *Let Φ be a formula of size s and formal degree at most d. Then there exist $k \in \mathbb{N}$ polynomials $g_1, \ldots, g_k$, $h_1, \ldots, h_k$ and a formula Φ' such that the following are true.*

- *The degree of each g_i is at least $\lfloor d/3 \rfloor$ and at most $2\lfloor d/3 \rfloor - 1$ and each h_i has degree at least 1.*

- *Φ' has formal degree at most $2\lfloor d/3 \rfloor$ and size at most s.*
- *The constant term of each g_i and h_i is zero.*
- *There exists a constant $c \in \mathbb{F}$ such that*

$$\Phi \equiv \Phi' + \sum_{i=1}^{k} g_i h_i + c\,.$$

- $k\lfloor d/3 \rfloor \leq s$.

Proof. For brevity, let us assume that d is divisible by 3, and let $t = d/3$. The proof of the lemma essentially follows from a repeated application of Lemma 3.3.3, where we keep extracting vertices of degree at least t till the total degree becomes smaller than $2t$. We now give the details.

Since the formal degree of Φ is at least $2t$, from Lemma 3.3.3, we know that there is a vertex v_1 in Φ with formal degree in the interval $[t, 2t-1]$ and polynomials h'_1 and f_1 satisfying the following properties.

- $\Phi \equiv h'_1 \cdot \Phi_{v_1} + f_1$.
- For every $\gamma \in \mathbb{F}$, $\gamma h'_1 + f_1$ can be computed by a formula of size at most $|\Phi| - |\Phi_{v_1}|$.

Let $g'_1 = \Phi_{v_1}$ and $s_1 = |\Phi_{v_1}|$. Let $\alpha, \beta \in \mathbb{F}$ be the constant terms of g'_1 and h'_1 respectively, and let g_1 and h_1 be polynomials with constant term zero such that $g'_1 = g_1 + \alpha$ and $h'_1 = h_1 + \beta$. We know that $\Phi \equiv h'_1 \cdot g'_1 + f_1$, or in other words, $\Phi \equiv (h_1 + \beta) \cdot (g_1 + \alpha) + f_1$. Rearranging the terms, we get

$$\Phi \equiv g_1 h_1 + \beta(g_1 + \alpha) + \alpha(h_1 + \beta) + f_1 - \alpha\beta\,.$$

This can be re-written as

$$\Phi \equiv g_1 h_1 + \beta g'_1 + (\alpha h'_1 + f_1) - \alpha\beta\,.$$

Observe that $\alpha h'_1 + f_1$ can be computed by a formula Φ_1 of size at most $s - s_1$. Also, the constant terms of g_1, h_1 are both zero and $\beta g'_1$ is computable by a formula $\Phi'_{v_1} \equiv \beta \cdot \Phi_{v_1}$ of formal degree at most $2t-1$ and size s_1 (which is also a sub-formula of Φ). In a nutshell, in one application of Lemma 3.3.3, we have decomposed Φ into a sum of a formula Φ'_{v_1} of size s_1 and formal degree at most $2t-1$, a formula Φ_1 of size at most $s - s_1$, a constant term and a product of two constant free polynomials $g_1 h_1$, where g_1 has degree at least t and h_1 has degree at least 1. To see the lower

bound on the degree of h_1, note that if h_1 is of degree zero, then it must be identically zero (since it is constant free), and this term just vanishes.

Now, consider the formula Φ_1. If the formal degree of Φ_1 is at most $2t-1$, then we are already done by letting Φ' in the statement of the theorem be the sum of Φ_1 and Φ'_{v_1} (indeed, the size of Φ' is at most $(s-s_1)+s_1 = s$). Else, observe that the size of Φ_1 is strictly smaller than the size of Φ. All the items of the lemma (except the last item) now follow from a simple induction on the formula size.

To see the upper bound on k given by the last item, note that the formulas for the polynomials $g'_1, g'_2, \ldots, g'_k$ obtained in each of the steps, whose sizes are $s_1, \ldots, s_k$, respectively, are all disjoint subformulas of Φ and have formal degree at least t each. Thus, by Observation 3.3.2, $s_i \geq t$ for all $i \in [k]$, and hence, the size of Φ is at least $kt = k\lfloor d/3 \rfloor$. □

A Quadratic Lower Bound Against Formulas

We are now ready to prove the lower bound on the formula size computing the elementary symmetric polynomial $\mathrm{ESYM}_{,(n,0.1n)}(\mathbf{x})$. As was the case with ABPs, before we prove Theorem 3.3.1, we need to prove the lower bound for formulas whose formal degree is at most $0.1n$.

Theorem 3.3.5. *Let $n \in \mathbb{N}$, and let $\mathbb{F}$ be a field of characteristic greater than $0.1n$. Let $A_1(\mathbf{x}), \ldots, A_r(\mathbf{x}), B_1(\mathbf{x}), \ldots, B_r(\mathbf{x})$ and $R(\mathbf{x})$ be polynomials such that for every i, $A_i(\mathbf{0}) = B_i(\mathbf{0}) = 0$ and R is a polynomial of degree at most $0.1n-1$. Then, for every $r \leq 0.1n$, any formula over $\mathbb{F}$, of formal degree at most $0.1n$ which computes the polynomial*

$$\mathrm{ESYM}_{,(n,0.1n)}(\mathbf{x}) + \sum_{j=1}^{r} A_j \cdot B_j + R$$

has at least $0.001n^2$ vertices.

Proof. Let $\mathcal{F}$ be a size s formula that computes a polynomial of the form

$$\mathrm{ESYM}_{,(n,0.1n)}(\mathbf{x}) + \sum_{j=1}^{r} A_j \cdot B_j + R$$

where $A_1(\mathbf{x}), \ldots, A_r(\mathbf{x}), B_1(\mathbf{x}), \ldots, B_r(\mathbf{x})$ and $R(\mathbf{x})$ are as in the statement of the lemma.

By Lemma 3.3.4, we know that the polynomial computed by $\mathcal{F}$ has the form

$$R' + \sum_{i=1}^{k} g_i h_i + c$$

where R' has degree at most $2\lfloor 0.1n/3 \rfloor \leq 0.1n - 1$, the constant terms of g_i and h_i are zero, and $s \geq k \cdot \lfloor 0.1n/3 \rfloor$.

Thus,

$$\begin{aligned} &\mathrm{ESYM}_{,(n,0.1n)}(\mathbf{x}) + \sum_{j=1}^{r} A_j \cdot B_j + R = R' + \sum_{i=1}^{k} g_i h_i + c \\ &\qquad \implies \mathrm{ESYM}_{,(n,0.1n)}(\mathbf{x}) + R'' = \sum_{i=1}^{k} g_i h_i - \sum_{j=1}^{r} A_j \cdot B_j \end{aligned}$$

where $R'' = R - R' - c$ has degree at most $0.1n - 1$. This shows that if we consider

$$V = \mathbb{V}\left(\left\{\partial_{x_i} \mathrm{ESYM}_{,(n,0.1n)}(\mathbf{x}) - \partial_{x_i} R'' \ : \ i \in [n]\right\}\right)$$

and

$$V' = \mathbb{V}\left(\{g_i, h_i, A_j, B_j \ : \ i \in [k], j \in [r]\}\right),$$

then $V' \subseteq V$ and $V' \neq \emptyset$.

Now by Lemma 3.2.5, we know that $\dim(V) \leq 0.1n - 2$. Therefore, since $V' \neq \emptyset$, $\dim(V') \geq n - 2k - 2r$ (see, for example, [Smi14, Section 2.8]). Hence,

$$\begin{aligned} &n - 2k - 2r \leq 0.1n - 2 \\ &\qquad \implies k \geq \frac{n - 0.1n - 2r + 2}{2} = 0.45n - r + 1 \geq 0.35n. \end{aligned}$$

This implies that $s \geq 0.35n \cdot \left\lfloor \frac{0.1n}{3} \right\rfloor \geq \frac{0.035n^2}{6} \geq 0.001n^2$. □

We now use this to prove a claim similar to Claim 2.4.13.

Claim 3.3.6. *Let $\mathcal{F}_k$ be a formula with formal degree $d_k > 0.1n$ and size at most τ, that computes a polynomial of the form*

$$\mathrm{ESYM}_{,(n,0.1n)}(\mathbf{x}) + \sum_{j=1}^{r_k} A_j \cdot B_j + R$$

where $A_1(\mathbf{x}), \ldots, A_r(\mathbf{x})$, $B_1(\mathbf{x}), \ldots, B_r(\mathbf{x})$ *are polynomials such that for every* j, $A_j(\mathbf{0}) = B_j(\mathbf{0}) = 0$ *and* $R(\mathbf{x})$ *is a polynomial of degree at most* $0.1n - 1$.

If $\tau \leq 0.001n^2$, *then there exists a formula* $\mathcal{F}_{k+1}$ *with formal degree at most* $2d_k/3$ *and size at most* τ *which computes a polynomial of the form*

$$\mathrm{ESYM}_{,(n,0.1n)}(\mathbf{x}) + \sum_{j=1}^{r_{k+1}} A'_j \cdot B'_j + R' \,,$$

where $A'_1(\mathbf{x}), \ldots, A'_{r_{k+1}}(\mathbf{x})$, $B'_1(\mathbf{x}), \ldots, B'_{r_{k+1}}(\mathbf{x}), R'(\mathbf{x})$ *are polynomials such that for every* i, $A'_i(\mathbf{0}) = B'_i(\mathbf{0}) = 0$, $R'(\mathbf{x})$ *is a polynomial of degree at most* $0.1n - 1$ *and* $r_{k+1} \leq r_k + \frac{0.0033n^2}{d_k}$.

Proof. Let $\mathcal{F}_k$ be a formula computing a polynomial of the form $\mathrm{ESYM}_{,(n,0.1n)}(\mathbf{x}) + \sum_{j=1}^{r_k} A_j \cdot B_j + R$, with formal degree $d_k > 0.1n$ and size at most τ. Here $R(\mathbf{x})$ has degree at most $0.1n - 1$ and $A_1(\mathbf{x}), \ldots, A_r(\mathbf{x}), B_1(\mathbf{x}), \ldots, B_r(\mathbf{x})$ are polynomials such that for every j, $A_j(\mathbf{0}) = B_j(\mathbf{0}) = 0$. Then, for $\Phi = \mathcal{F}_k$, Lemma 3.3.4 says that there exists a formula $\mathcal{F}_{k+1} = \Phi'$ of formal degree at most $2d_k/3$ and size at most τ that computes a polynomial of the form

$$\mathrm{ESYM}_{,(n,0.1n)}(\mathbf{x}) + \sum_{j=1}^{r_k} A_j \cdot B_j + R - \sum_{i=1}^{\ell_k} g_i \cdot h_i + c \,,$$

where $\ell_k \leq \frac{\tau}{\lfloor d_k/3 \rfloor} \leq \frac{3.3 \times 0.001n^2}{d_k} = \frac{0.0033n^2}{d_k}$ for large enough n. Further, $g_1, \ldots, g_{\ell_k}$, $h_1, \ldots, h_{\ell_k}$ are a set of non-constant polynomials with constant term zero and $c \in \mathbb{F}$. Thus, if we set $R' = R + c$, $A'_i = A_i, B'_i = B_i$ for every $i \in [r_k]$ and $A'_{r_k+j} = g_j, B'_{r_k+j} = h_j$ for every $j \in [\ell_k]$, the claim follows. □

Finally, let us recall the main theorem of this chapter and complete its proof.

Theorem 3.3.7. *Let* $n \in \mathbb{N}$ *and let* $\mathbb{F}$ *be a field of characteristic greater than* $0.1n$. *Then any algebraic formula over* $\mathbb{F}$ *computing the elementary symmetric polynomial*

$$\mathrm{ESYM}_{,(n,0.1n)}(\mathbf{x}) = \sum_{S \subseteq [n], |S| = 0.1n} \prod_{j \in S} x_j \,,$$

is of size at least $\Omega(n^2)$.

Proof. Let $\mathcal{F}$ be a formula with formal degree d_0 which computes the polynomial $\mathrm{ESYM}_{,(n,0.1n)}(\mathbf{x})$. We assume without loss of generality that the underlying field $\mathbb{F}$ is algebraically closed. If d_0 is at most $0.1n$, then by Theorem 3.3.5, we know that $|\mathcal{F}|$

is at least $\Omega(n^2)$ and we are done. Also, if $d_0 > n^2$, then we again have the required lower bound by Observation 3.3.2. Thus, we may assume that $0.1n \leq d_0 \leq n^2$.

From this point, the proof is exactly along the same lines as that of Theorem 2.2.1. We iteratively make changes to $\mathcal{F}$, reducing its formal degree geometrically in each step, via Lemma 3.3.4, till we get a formula $\mathcal{F}'$ of formal degree at most $0.1n$ that computes a polynomial of the form

$$\mathrm{ESYM}_{,(n,0.1n)}(\mathbf{x}) + \sum_{j=1}^{r} A_j \cdot B_j + R$$

where $r \leq 0.1n$ and $A_1(\mathbf{x}), \ldots, A_r(\mathbf{x}), B_1(\mathbf{x}), \ldots, B_r(\mathbf{x})$ are polynomials such that for every i, $A_i(\mathbf{0}) = B_i(\mathbf{0}) = 0$ and $R(\mathbf{x})$ is a polynomial of degree at most $0.1n - 1$. Once we have this, Theorem 3.3.5 gives the required lower bound.

As before, we use Claim 3.3.6 iteratively to obtain $\mathcal{F}'$ from $\mathcal{F}$. Set $\mathcal{F}_0 = \mathcal{F}$. Then, $\mathcal{F}_0$ is a formula with formal degree d_0 and size at most τ that computes the polynomial $\mathrm{ESYM}_{,(n,0.1n)}(\mathbf{x})$. If $\tau \geq 0.001n^2$, the statement of the theorem follows.

Otherwise, applying Claim 3.3.6 iteratively as long as the formal degree remains $> 0.1n$, we eventually get a formula $\mathcal{F}'$ with formal degree $d' \leq 0.1n$. Let the number of steps taken be K, and so let $\mathcal{F}_K = \mathcal{F}'$. Further, let $d_0, \ldots, d_{K-1}, d_K$ denote the formal degree of each formula in this sequence. Thus $d_{K-1} > 0.1n$, and $d_k \leq 2d_{k-1}/3$ for every $k \in [K]$. Now $\mathcal{F}'$ has formal degree at most $0.1n$, size at most τ, and computes a polynomial of the form

$$\mathrm{ESYM}_{,(n,0.1n)}(\mathbf{x}) + \sum_{j=1}^{r_K} A_j \cdot B_j + R\,,$$

where $A_1(\mathbf{x}), \ldots, A_r(\mathbf{x})$, $B_1(\mathbf{x}), \ldots, B_r(\mathbf{x})$ are polynomials such that for every i, $A_i(\mathbf{0}) = B_i(\mathbf{0}) = 0$ and $R(\mathbf{x})$ is a polynomial of degree at most $0.1n - 1$. Further, the number of error terms, r_K, is at most

$$0.0033n^2 \left(\frac{1}{d_{K-1}} + \frac{1}{d_{K-2}} + \cdots + \frac{1}{d_0} \right).$$

Since $d_k \leq \frac{2}{3} \cdot d_{k-1}$, we have that $\frac{1}{d_{k-1}} \leq \frac{2}{3} \cdot \frac{1}{d_k}$ for all $k \in [K]$. This implies that

$$r_K \leq 0.0033n^2 \cdot \frac{1}{1 - 2/3} \cdot \frac{1}{d_{K-1}} = \frac{0.0099n^2}{0.1n} = 0.099n \leq 0.1n$$

as $d_{K-1} \geq 0.1n$.

Finally, since the formal degree is at most $0.1n$, using Theorem 3.3.5 we get

$$\tau \geq \left|\mathcal{F}'\right| \geq 0.001n^2 = \Omega(n^2)\,. \qquad \square$$

We conclude with an open problem. A question which is natural in the context of this work and remains open is to prove super-quadratic lower bounds for general formulas. As a first step towards this, the question of proving super-quadratic lower bound for homogeneous algebraic formulas might be more approachable.

4 The Non-Commutative Setting: Circuit Lower Bounds

In this and the following chapter, we are interested in polynomials that come from the non-commutative polynomial ring $\mathbb{F}\langle x_1, \ldots, x_n \rangle$, where the indeterminates do not commute with each other (that is, $xy \neq yx$ for indeterminates x, y). As a consequence, any monomial in a non-commutative polynomial $f \in \mathbb{F}\langle x_1, \ldots, x_n \rangle$ is essentially a string over the alphabet $\{x_1, \ldots, x_n\}$. Such models occur naturally while studying computations on matrices, among other things. There has been a long line of work that studies non-commutative computation beginning with the works of Hyafil and Nisan [Hya77; Nis91][1].

It was shown by Hrubeš, Wigderson and Yehudayoff [HWY10] that the non commutative permanent polynomial is complete for the class $\mathsf{VNP}_{\mathrm{nc}}$ (the non-commutative version of VNP). Later Arvind, Joglekar and Raja [AJR16] gave a natural polynomial that is complete for the class of n-variate non-commutative polynomials computable by $\mathrm{poly}(n)$-sized circuits (denoted by $\mathsf{VP}_{\mathrm{nc}}$).

The question of whether the classes $\mathsf{VP}_{\mathrm{nc}}$ and $\mathsf{VNP}_{\mathrm{nc}}$ are different is the central open problem in the non-commutative setting. Although the general question of showing lower bounds against non-commutative circuits remains open, there has been considerable progress in restricted settings [LMS16; LMP19; LLS19; ST18; Fij+20; LST21a]. Most significantly, Limaye, Srinivasan and Tavenas [LST21a] gave a super polynomial lower bound against constant depth non-commutative circuits. They showed that any depth-Δ circuit computing the iterated matrix multiplication polynomial, $\mathrm{IMM}_{n,d}(\mathbf{x})$, must have size at least $n^{\Omega(d^{1/\Delta})}$ when Δ is constant.

With respect to the general question, Hrubeš, Wigderson and Yehudayoff [HWY11] showed that a sufficiently strong super-linear lower bound for the classical sum-of-squares problem implies a separation between $\mathsf{VP}_{\mathrm{nc}}$ and $\mathsf{VNP}_{\mathrm{nc}}$. In another related work, Carmosino, Impagliazzo, Lovett and Mihajlin [Car+18] showed that proving mild lower bounds against non-commutative circuits would imply exponential lower bounds against the same model.

[1] The main result in [Hya77] is unfortunately false as shown in [Nis91]

One motivation for studying non-commutative computation is that it is possibly easier to prove strong lower bounds in this setting as compared to the usual commutative setting. At least intuitively, it seems harder to *cancel* monomials once they have been calculated when commutativity is not allowed amongst the variables. For example, the $n \times n$ determinant can be computed by an $O(n^3)$ algebraic circuit, but to the best of our knowledge there is no circuit for the non-commutative determinant of size $2^{o(n)}$. In fact, it was shown by Arvind and Srinivasan [AS18] that if the non-commutative determinant has a poly-sized circuit, then $\mathsf{VP}_{\mathrm{nc}} = \mathsf{VNP}_{\mathrm{nc}}$.

Even though super-polynomial lower bound against non-commutative circuits are not known, exponential lower bounds are known in the non-commutative setting for formulas and algebraic branching programs. We will talk about these in the next chapter. In this chapter, we define a new class of non-commutative circuits that allow us to get strong lower bounds against general non-commutative circuits from weak lower bounds against multilinear circuits from the commutative world.

4.1 Abecedarian Polynomials and Circuits

In [HWY11], Hrubeš *et al.* defined the notion of *ordered* polynomials. A homogeneous polynomial of degree d is said to be ordered if the set of variables it depends on can be partitioned into d buckets such that in every monomial, variables occurring in position k only come from the k-th bucket.

We generalise this notion by making the bucket indices *position independent*. That is, a variable in position k need not necessarily come from the k-th bucket as long as in every monomial, the variables appear in non-decreasing order of their bucket indices. We call such polynomials abecedarian since, in English, an abecedarian word is one in which all of the letters are arranged in alphabetical order [MW19].

The difference between ordered polynomials and abecedarian ones can be explained succinctly using the notion of *regular expressions* from Automata Theory.

For a non-commutative polynomial $f \in \mathbb{F}\langle x_1, \ldots, x_n \rangle$, suppose the variables can be partitioned into buckets $\{X_1, \ldots, X_m\}$. Then f is said to be *ordered* with respect to $\{X_1, \ldots, X_m\}$ if every monomial in it is a word that can be generated using the *regular expression* $X_1 \cdots X_m$. On the other hand, f is abecedarian if the monomials in it are words that can be generated using the regular expression $X_1^* \cdots X_m^*$.

It is easy to see that any ordered polynomial is also abecedarian with respect to the same partition. This is because position indices are always increasing. For example, consider the following version of the *complete homogeneous symmetric polynomial*.

$$\mathrm{CHSYM}^{(\mathsf{ord})}_{n,d}(\mathbf{x}) = \sum_{1 \le i_1 \le \ldots \le i_d \le n} x^{(1)}_{i_1} \cdots x^{(d)}_{i_d}.$$

It is both ordered, as well as abecedarian with respect to the same partition – $\left\{X_k = \left\{x_i^{(k)} \ : \ i \in [n]\right\}\right\}$.

On the other hand, note that there are homogeneous abecedarian polynomials that are not ordered. The following version of the same polynomial is an example.

$$\mathrm{CHSYM}_{n,d}(\mathbf{x}) = \sum_{1 \le i_1 \le \ldots \le i_d \le n} x_{i_1} \cdots x_{i_d}$$

is abecedarian with respect to $\{X_i \ : \ X_i = \{x_i\}\}$, but is not ordered.

The reason is that for a polynomial to be ordered, the bucket labels have to essentially be position labels. On the other hand, for a polynomial to be abecedarian with respect to a partition, the bucket labels can be independent of position.

We now formally define abecedarian polynomials.

Definition 4.1.1 (Abecedarian Polynomials)**.** *Let $f \in \mathbb{F}\langle x_1, \ldots, x_n \rangle$ be a polynomial of degree d and $\{X_1, \ldots, X_m\}$ be a partition for $\{x_1, \ldots x_n\}$. Further, for any $k \in [d]$, $f[X_{i_1}, \ldots, X_{i_k}]$ is defined as follows.*

For a polynomial f, $f[X_{i_1}, \ldots, X_{i_k}]$ is the homogeneous polynomial of degree k such that for every monomial α,

$$\mathsf{coeff}_\alpha(f[X_{i_1}, \ldots, X_{i_k}]) = \begin{cases} \mathsf{coeff}_\alpha(f) & \text{if } \alpha = x_{\ell_1} \cdots x_{\ell_k} \text{ with } x_{\ell_j} \in X_{i_j} \text{ for every } j \in [k] \\ 0 & \text{otherwise.} \end{cases}$$

Then f is said to be abecedarian *with respect to a partition, if*

$$f = f[\emptyset] + \sum_{k=1}^{d} \left(\sum_{1 \le i_1 \le \cdots \le i_k \le m} f[X_{i_1}, \ldots, X_{i_k}] \right)$$

where $f[\emptyset)$ is the constant term in f.

In this case, we say that f is abecedarian *with respect to $\{X_1, \ldots, X_m\}$, a partition of size m.* $\diamondsuit$

Before moving ahead, let us go over a few subtleties to help us understand the subclass of abecedarian polynomials better.

Examples and Non-Examples Firstly we note that the most natural way of considering the non-commutative analogue of *any* commutative polynomial yields an abecedarian polynomial with respect to the partition $\{X_i \ : \ X_i = \{x_i\}\}$. That is, given a commutative polynomial $f \in \mathbb{F}[x_1, \ldots, x_n]$, if we define its non-commutative analogue, $f^{(\mathrm{nc})}$ as follows.

> f and $f^{(\mathrm{nc})}$ look essentially the same, except that variables in every monomial in $f^{(\mathrm{nc})}$ are arranged in non-decreasing order of their indices.

Then, $f^{(\mathrm{nc})}$ is abecedarian with respect to the partition $\{X_i \ : \ X_i = \{x_i\}\}$.

However, we note that the natural non-commutative analogues of commutative computation need not be abecedarian. For example, the ABP constructed by Mahajan and Vinay [MV97] to compute the determinant is not abecedarian even though the usual ABP (see, for example, [LMP19, Lemma 15]) used to compute the elementary symmetric polynomial is.

This leads to the fact that not every non-commutative analogue of a commutative polynomial is abecedarian with respect to the partition $\{X_i \ : \ X_i = \{x_i\}\}$.

For instance, consider the *arc-full rank polynomial* which was constructed by Dvir, Malod, Perifel and Yehudayoff to give a super-polynomial separation between the powers of formulas and ABPs in the multilinear setting [Dvi+12, Section 3.1]. Let us call that polynomial f and look at it as a non-commutative polynomial, f', in the following sense.

> Let $\mathcal{A}$ be the ABP that computes f and think of $\mathcal{A}$ as a non-commutative ABP $\mathcal{A}'$. Then, f' is the polynomial computed by $\mathcal{A}'$.

We note that across different monomials in f', the order in which variables appear is not consistent. Thus, f' is not abecedarian with respect to the given partition.

Polynomials Can be Abecedarian With Respect to Different Partitions A final point to note before we move ahead is that a polynomial might be abecedarian with respect to different partitions[2]. For example, note that $\mathrm{CHSYM}^{(\mathsf{ord})}_{n,d}(\mathbf{x})$ is abecedarian with respect to the partition $\left\{X_i = \left\{x_i^{(k)} \ : \ k \in [d]\right\}\right\}$ as well as $\left\{X_k = \left\{x_i^{(k)} \ : \ i \in [n]\right\}\right\}$. In fact, note that even the sizes of the different partitions are different.

Similarly, the polynomial

$$\mathrm{ESYM}^{(\mathsf{ord})}_{n,d}(\mathbf{x}) = \sum_{1 \le i_1 < \ldots < i_d \le n} x_{i_1}^{(1)} \cdots x_{i_d}^{(d)}$$

is abecedarian with respect to the partition $\left\{X_k = \left\{x_i^{(k)} \ : \ i \in [n]\right\}\right\}$ which has size d, as well as the partition $\left\{X_i = \left\{x_i^{(k)} \ : \ k \in [d]\right\}\right\}$ which has size n.

Next, we define a subclass circuits that naturally compute abecedarian polynomials.

Abecedarian Circuits

In the same paper that introduced ordered polynomials [HWY11], Hrubeš *et al.* also defined *ordered circuits*, a model that naturally computes ordered polynomials. We generalise this notion to define abecedarian circuits in the most natural way.

A circuit, $\mathcal{C}$, is abecedarian if every gate in $\mathcal{C}$ computes an abecedarian polynomial.

In their paper Hrubeš *et al.* had also shown that without loss of generality, any circuit computing an ordered polynomial can be assumed to be ordered [HWY11, Theorem 7.1]). We show that an analogous statement is true in the abecedarian setting.

Theorem 4.1.2 (Converting Circuits into Abecedarian Circuits)**.** *Suppose f is an* abecedarian *polynomial with respect to a partition of size m, and $\mathcal{C}$ is a circuit of size s computing f. Then there is an* abecedarian *circuit $\mathcal{C}'$ computing f of size $O(m^3 s)$.*

Before we move on to the proof, let us first define some notation.

Definition 4.1.3 (Sub-Polynomials of an Abecedarian Polynomial)**.** *Suppose f is an* abecedarian *polynomial with respect to the partition $\{X_1, \ldots, X_m\}$, and has degree d. For any $1 \le a \le b \le m+1$, $f[a,b)$ is the sub-polynomial of f defined as follows.*

- *For any $a \in [m+1]$, $f[a,a) = f[\emptyset)$ is the constant term in f.*

[2] Every polynomial $f \in \mathbb{F}\langle x_1, \ldots, x_n \rangle$ is abecedarian with respect to the partition $\{X\}$ for $X = \{x_1, \ldots, x_n\}$.

- *For any* $1 \leq a < b \leq m+1$,

$$f[a,b) = \sum_{k=1}^{d} \left(\sum_{\substack{i_1,\ldots,i_k \in [m] \\ a=i_1 \leq \cdots \leq i_k < b}} f[X_{i_1}, \ldots, X_{i_k}] \right)$$

where $f[X_{i_1}, \ldots, X_{i_k}]$ *is as defined in Definition 4.1.1.* $\Diamond$

That is, given a polynomial f that is abecedarian with respect to the partition $\{X_1, \ldots, X_m\}$, we will use $f[a,b)$ to denote the polynomial that consists of all those monomials in f that can be generated by the regular expression $X_a^+ X_{a+1}^* \cdots X_{b-1}^*$.

Proof of Theorem 4.1.2. Without loss of generality, let us assume that the $\mathcal{C}$ has fan-in 2. We prove the given statement by describing how to construct $\mathcal{C}'$ from $\mathcal{C}$.

For each gate v in $\mathcal{C}$, we make $O(m^2)$ copies in $\mathcal{C}'$, $\{(v,[a,b)) \ : \ 1 \leq a \leq b \leq m+1\}$. Further, if root is the output gate in $\mathcal{C}$, then we define a new output gate in $\mathcal{C}'$ that computes $\sum_{i=1}^{m} (\text{root}, [i, m+1))$.

Intuitively, if f_v is the polynomial computed at v in $\mathcal{C}$, then the polynomial computed at $(v,[a,b))$ is $f_v[a,b)$. Thus if f was the polynomial computed at root, then the polynomial computed by $\mathcal{C}'$ is $\sum_{i=1}^{m+1} f[i, m+1)$ which is indeed f.

We ensure this property at every gate by adding edges as follows.

- If v is an input gate labelled by a field element γ,
  - we set $(v,[a,a)) = \gamma$ for every $a \in [m+1]$;
  - we set $(v,[a,b)) = 0$ for every $1 \leq a < b \leq m+1$.
- If v is an input gate labelled by a variable x_i and $x_i \in X_k$,
  - we set $(v,[k,k+1)) = x_i$;
  - we set $(v,[a,b)) = 0$ for every $a \neq k$, $b \neq k+1$.
- If $v = v_1 + v_2$, we set

$$(v,[a,b)) = (v_1,[a,b)) + (v_2,[a,b))$$

for every $a \leq b \in [m+1]$.

- If $v = v_1 \times v_2$, we set

$$(v, [a, a)) = (v_1, [a, a)) \cdot (v_2, [a, a))$$

for every $a \in [m + 1]$; and

$$(v, [a, b)) = (v_1, [a, a)) \cdot (v_2, [a, b)) + (v_1, [a, b)) \cdot (v_2, [b, b)) + \sum_{c=a}^{b-1} (v_1, [a, c + 1)) \times (v_2, [c, b))$$

for every $1 \leq a < b \leq m + 1$.

Using induction, it is easy to see[3] that every gate in $\mathcal{C}'$ computes the intended abecedarian polynomial. Further for every gate v in $\mathcal{C}$, there are at most $O(m^3)$ vertices in $\mathcal{C}'$. Therefore, $\mathcal{C}'$ is indeed an abecedarian circuit computing f of size $O(m^3 s)$. □

Note that Theorem 4.1.2 implies that in order to show super-polynomial lower bounds against general non-commutative circuits, it is enough to show super-polynomial lower bounds against abecedarian circuits. What we will see now is a way to show strong lower bounds against non-commutative circuits using weak lower bounds against *multilinear* circuits in the commutative world.

4.2 Non-Commutative Circuit Lower Bounds From Multilinear Circuit Lower Bounds

In this section, we will show that strong enough lower bounds against commutative multilinear circuits can be *amplified* to give arbitrarily strong polynomial lower bounds against general non-commutative circuits.

In particular, we will prove the following statement.

Theorem 4.2.1. *For any $\varepsilon > 0$, assume that there exists an explicit n-variate commutative multilinear polynomial of degree $\mathrm{poly}(n)$, such that any multilinear circuit computing it requires size $\Omega(n^{3+(\omega/2)+\varepsilon})$. Then for any $c > 1$, there exists another explicit m-variate polynomial of degree $\mathrm{poly}(m)$, such that any non-commutative circuit computing it requires size $\Omega(m^c)$.*

[3] Essentially, *parse trees* ([LLS19]) of $\mathcal{C}$ are simply rearranged in $\mathcal{C}'$ and therefore each parse tree in $\mathcal{C}$ appears exactly once in $\mathcal{C}'$.

Even though the statement seems surprising, it follows rather easily from techniques similar to those in the proof of Theorem 4.1.2 (or the results in the work of Hrubeš *et al.* [HWY11, Section 8]) and one of the main results in the work of Carmosino *et al.* [Car+18, Theorem 1.1]. We give a proof for the sake of completeness.

Let us first state the *hardness amplification* statement from [Car+18], where it was shown that super-linear lower bounds against general non-commutative circuits are enough to show arbitrarily strong polynomial lower bounds against the same model.

Theorem 4.2.2 (Theorem 1.1 in [Car+18])**.** *For any $\varepsilon > 0$, assume that there exists an explicit non-commutative polynomial in n variables of degree $\mathrm{poly}(n)$, such that any non-commutative circuit computing it requires size $\Omega(n^{(\omega/2)+\varepsilon})$. Then, for any $c > 1$, there exists another explicit polynomial in m variables of degree $\mathrm{poly}(m)$, such that any non-commutative circuit computing it requires size $\Omega(m^c)$.*

We now prove Theorem 4.2.1

Proof of Theorem 4.2.1. Let $\varepsilon > 0$ be fixed, and $f \in \mathbb{F}[x_1, \ldots, x_n]$ be the explicit multilinear polynomial in n variables of degree $\mathrm{poly}(n)$, such that any multilinear circuit computing it has size at least $\gamma \cdot n^{3+(\omega/2)+\varepsilon}$ for some absolute constant γ.

We define the non-commutative analogue of f, say $f^{(\mathrm{nc})}$ in the following way.

> f and $f^{(\mathrm{nc})}$ look essentially the same, except that variables in every monomial in $f^{(\mathrm{nc})}$ are arranged in non-decreasing order of their indices.

We then claim the following.

Claim 4.2.3. *If $\mathcal{C}$ is a non-commutative circuit computing $f^{(\mathrm{nc})}$ of size s, then there is a multilinear circuit $\mathcal{C}'$ computing f of size at most $\delta \cdot n^3 s$ for some absolute constant δ.*

Before proving the claim, let us complete the proof using it.

Suppose $\mathcal{C}$ is a non-commutative circuit of size s computing f^{nc}. Then, by Claim 4.2.3, we have that there is a multilinear circuit $\mathcal{C}'$ that computes f of size at most $\delta \cdot n^3 s$ for some absolute constant δ. By our assumption, $\delta \cdot n^3 s \geq \gamma \cdot n^{3+(\omega/2)+\varepsilon}$, and so $s \geq (\gamma/\delta) \cdot n^{(\omega/2)+\varepsilon}$.

Therefore, we have an explicit n-variate non-commutative polynomial, f, of degree $\mathrm{poly}(n)$, such that any non-commutative circuit computing it requires size $\Omega(n^{(\omega/2)+\varepsilon})$. By Theorem 4.2.2, this implies that for any $c > 1$, there exists another

explicit polynomial in m variables of degree $\mathrm{poly}(m)$, such that any non-commutative circuit computing it requires size $\Omega(m^c)$.

To complete the proof, we therefore need to prove Claim 4.2.3, whose proof closely follows that of Theorem 4.1.2.

Proof of Claim 4.2.3. We prove the claim by describing how to construct $\mathcal{C}'$ from $\mathcal{C}$. For each gate v in $\mathcal{C}$, let us make $O(n^2)$ copies of v in $\mathcal{C}'$, namely

$$\left\{v_{[a,b)} \ : \ 1 \leq a < b \leq n+1\right\} \cup \{v_\emptyset\}\,.$$

Further, if root is the output gate in $\mathcal{C}$, then we define a new output gate in $\mathcal{C}'$ that computes $\sum_{i=1}^{n}(\mathsf{root}, [i, n+1))$.

Intuitively, suppose g_v is the polynomial computed at v in $\mathcal{C}$. Then the polynomial computed at $v_{[a,b)}$, say g'_v, will be the sum of those multilinear monomials (along with their coefficients) of g_v for which the following is true.

> The lowest index among the indices of variables present in the monomial is exactly a, and the highest index among the indices of variables present it is at most (and including) $b-1$.

Thus if f was the polynomial computed at root, then the polynomial computed by $\mathcal{C}'$ is $\sum_{i=1}^{n+1} f[i, n+1)$ which is indeed f.

The polynomial computed at $v_\emptyset$ is the constant term of g_v. The way we ensure this property at every gate is by building the circuit C' bottom-up as follows.

- If v is an input gate labelled by a field element α, set $v_\emptyset$ to be α and $v_{[a,b)} = 0$ for every $1 \leq a < b \leq n+1$.
- If v is an input gate labelled by a variable x_i, then set $v_{[i,i+1)} = x_i$ and $v_{[a,b)} = 0$ for all other $1 \leq a < b \leq n+1$. Also set $v_\emptyset = 0$.
- If $v = u + u'$, set $v_\emptyset = u_\emptyset + u'_\emptyset$ and

  $$v_{[a,b)} = u_{[a,b)} + u'_{[a,b)}$$

  for every $1 \leq a < b \leq n+1$.
- If $v = u \times u'$, set $v_\emptyset = u_\emptyset \times u'_\emptyset$. Also, assuming $[i,i) = \emptyset$ for any i, set

  $$v_{[a,b)} = \sum_{a \leq i \leq b} u_{[a,i)} \times u'_{[i,b)}$$

for every $1 \leq a < b \leq n+1$.

By induction, one can easily show that[4] the gates in $\mathcal{C}'$ indeed have the property that we claimed them to have before we described $\mathcal{C}'$. Hence the polynomial computed by C' is indeed f. Further, for every gate v in $\mathcal{C}$, there are at most $O(n^3)$ vertices in $\mathcal{C}'$, and so the size of $\mathcal{C}'$ is as claimed. □ □

Theorem 4.2.1 shows that a strong enough lower bound against multilinear circuits is enough to show arbitrarily strong polynomial lower bounds against non-commutative circuits. On the other hand, the best lower bound known against multilinear circuits is an almost quadratic lower bound due to Alon, Kumar and Volk [AKV20].

Therefore, it would be very interesting to know whether super-quadratic lower bounds against multilinear circuits implies arbitrarily strong polynomial lower bounds against non-commutative circuits.

[4] Essentially, *parse trees* ([LLS19]) of $\mathcal{C}$ are simply rearranged in $\mathcal{C}'$ and therefore each parse tree in $\mathcal{C}$ appears exactly once in $\mathcal{C}'$.

5

Separating Syntactically Abecedarian Formulas and Algebraic Branching Programs

As we saw in the previous chapter, for the case of circuits, we do not know a lower bound better than the one known in the commutative case itself, due to Baur and Strassen [Str73a; BS83]. However, for the case of formulas, we know an exponential lower bound due to the seminal work of Nisan [Nis91]. In fact, the proof actually works for Algebraic Branching Programs – a model which is believed to be more powerful than formulas. Nisan's work [Nis91] actually gives an exact characterisation for the size of any algebraic branching program (or ABP) computing a non-commutative polynomial.

The motivating question for this chapter is whether there is a separation between the powers of ABPs and formulas in the non-commutative setting. Let us denote the class of non-commutative polynomials over n variables that can be computed by $\mathrm{poly}(n)$-sized ABPs by $\mathsf{VBP}_{\mathrm{nc}}$. Similarly, let $\mathsf{VF}_{\mathrm{nc}}$ denote the class of non-commutative polynomials over n variables that can be computed by $\mathrm{poly}(n)$-sized formulas. The question is essentially whether $\mathsf{VBP}_{\mathrm{nc}}$ is contained in $\mathsf{VF}_{\mathrm{nc}}$.

This question had been posed by Nisan [Nis91] and at the time when this work was done, the only work we were aware of that made some progress with respect to this question was by Lagarde, Limaye and Srinivasan [LLS19], where they show that certain syntactically restricted non-commutative formulas (called Unique Parse Tree formulas) cannot compute $\mathrm{IMM}_{n,n}$ unless they have size $n^{\Omega(\log n)}$. However very recently, in a breakthrough work, Limaye, Srinivasan and Tavenas [LST21a] showed that in the homogeneous non-commutative setting, there is indeed a super-polynomial separation between the powers of formulas and ABPs.

In this chapter, we extend the definition of abecedarian circuits to define abecedarian ABPs (ABPs in which the polynomial computed between any two of its vertices is abecedarian) and abecedarian formulas (syntactically restricted formulas in which the polynomial computed at every vertex is abecedarian), and then show a super-polynomial separation between the powers of abecedarian formulas and ABPs.

5.1 Abecedarian Formulas and ABPs

Let us begin by recalling the definition of abecedarian polynomials. For a non-commutative polynomial $f \in \mathbb{F}\langle x_1, \ldots, x_n \rangle$, suppose the variables can be partitioned into buckets $\{X_1, \ldots, X_m\}$. Then, f is said to be abecedarian with respect to this partition if in every monomial, the variables appear in non-decreasing order of their bucket indices. A formal definition and further discussions can be found in section 4.1.

Given an abecedarian polynomial, we also defined its sub-polynomials as follows.

> Suppose f is an abecedarian polynomial with respect to the partition $\{X_1, \ldots, X_m\}$, and has degree d. For any $1 \leq a \leq b \leq m+1$, $f[a,b)$ is the sub-polynomial of f defined as follows.
>
> - For any $a \in [m+1]$, $f[a,a) = f[\emptyset)$ is the constant term in f.
> - For any $1 \leq a < b \leq m+1$,
>
> $$f[a,b) = \sum_{k=1}^{d} \left(\sum_{\substack{i_1,\ldots,i_k \in [m] \\ a=i_1 \leq \cdots \leq i_k < b}} f[X_{i_1}, \ldots, X_{i_k}] \right)$$
>
> where $f[X_{i_1}, \ldots, X_{i_k}]$ is the homogeneous polynomial of degree k such that for every monomial α, where the coefficient of α in $f[X_{i_1}, \ldots, X_{i_k}]$ is
>
> $$\begin{cases} \mathsf{coeff}_\alpha(f) & \text{if } \alpha = x_{\ell_1} \cdots x_{\ell_k} \text{ with } x_{\ell_j} \in X_{i_j} \text{ for every } j \in [k] \\ 0 & \text{otherwise.} \end{cases}$$

Finally, recall that we defined abecedarian circuits as non-commutative circuits in which every gate computes an abecedarian polynomial. However note that the abecedarian circuit, $\mathcal{C}'$, that we constructed given a general non-commutative circuit computing a polynomial f, in Theorem 4.1.2, had the following syntactic structure.

- $\mathcal{C}' = \mathcal{C}'_1 + \cdots + \mathcal{C}'_m$ where $\mathcal{C}'_i$ computed the polynomial $f[i, m+1)$;
- every gate v in $\mathcal{C}'_i$ was associated with a set $I_v = [a,b)$, and in particular, the root node in $\mathcal{C}'_i$ was associated with the set $[i, m+1)$
- if f_v was the polynomial computed at v, then $f_v = f_v[a,b)$;

- if $v = v_1 + v_2$, then $I_v = I_{v_1} = I_{v_2}$;
- if $v = v_1 \times v_2$ with $I_v = [a, a)$, then $I_{v_1} = I_{v_2} = [a, a)$
- if $v = v_1 \times v_2$ with $I_v = [a, b)$ and $a < b$, then one of the following was true
  - $I_{v_1} = [a, b)$ and $I_{v_2} = [b, b)$;
  - $I_{v_1} = [a, a)$ and $I_{v_2} = [a, b)$;
  - there exists $a \leq c < b$ such that $I_{v_1} = [a, c + 1)$ and $I_{v_2} = [c, b)$.

Let us call circuits that have this additional structure as syntactically abecedarian circuits. We note that Theorem 4.1.2 shows that up to polynomial factors, abecedarian circuits and syntactically abecedarian circuits are equivalent. However it is not clear if an analogous statement is true in the case of formulas.

Abecedarian Formulas

Similar to circuits, we can define abecedarian formulas to be those non-commutative formulas in which the polynomial computed at every gate is abecedarian, and syntactically abecedarian formulas as follows.

Definition 5.1.1 (Syntactically Abecedarian Formulas)**.** *For any $a, b \in \mathbb{N}$, let $[a, b)$ denote a set of the form $I = \{i \ : \ a \leq i < b\}$. Suppose $\mathcal{F}$ is a formula computing a polynomial f that is* abecedarian *with respect to a partition of size m. Then $\mathcal{F}$ is said to be syntactically* abecedarian *if $\mathcal{F} = \mathcal{F}_1 + \cdots + \mathcal{F}_m$ for sub-formulas $\mathcal{F}_1, \ldots, \mathcal{F}_{m+1}$, where for every $i \in [m + 1]$:*

- *$\mathcal{F}_i$ computes the polynomial $f[i, m + 1)$;*
- *every gate v in $\mathcal{F}_i$ is associated with a set $I_v = [a, b)$, and in particular, the root node must be associated with the set $[i, m + 1)$*
- *if f_v is the polynomial computed at v, then $f_v = f_v[a, b)$;*
- *if $v = v_1 + v_2$, then $I_v = I_{v_1} = I_{v_2}$;*
- *if $v = v_1 \times v_2$ with $I_v = [a, a)$, then $I_{v_1} = I_{v_2} = [a, a)$*
- *if $v = v_1 \times v_2$ with $I_v = [a, b)$ and $a < b$, then one of the following is true*
  - *$I_{v_1} = [a, b)$ and $I_{v_2} = [b, b)$;*
  - *$I_{v_1} = [a, a)$ and $I_{v_2} = [a, b)$;*

– *there exists* $a \leq c < b$ *such that* $I_{v_1} = [a, c+1)$ *and* $I_{v_2} = [c, b)$.

Further, $\mathcal{F}$ *is said to be homogeneous if each* $\mathcal{F}_i$ *is homogeneous.* $\Diamond$

As mentioned earlier, unlike circuits, it is not clear whether the syntactic and semantic definitions of abecedarian formulas are equivalent up to polynomial factors.

We now define abecedarian ABPs and show that any ABP computing an abecedarian polynomial can be converted into an abecedarian ABP with only a polynomial factor blow-up in size. It also follows easily that any polynomial which is computable by an efficient abecedarian formula is also computable by an efficient abecedarian ABP and similarly any polynomial which is computable by an efficient abecedarian ABP is also computable by an efficient abecedarian circuit.

Abecedarian Algebraic Branching Programs

An abecedarian algebraic branching program is a non-commutative ABP in which the polynomial computed between any two of its vertices is abecedarian. Let us begin by observing that similar to circuits, without loss of generality, we can assume that any ABP computing an abecedarian polynomial is abecedarian.

Observation 5.1.2 (Converting ABPs into Abecedarian ABPs)**.** *Suppose* f *is a degree* d abecedarian *polynomial with respect to a partition of size* m*. If there is an ABP* $\mathcal{A}_0$ *of size* s *computing it, then there is an* abecedarian *ABP* $\mathcal{A}'$ *computing it of size* $O(msd)$.

Proof. Firstly, we note that if f_0 is homogeneous, then there is a homogeneous[1] ABP $\mathcal{A}$ computing f_0 of size at most $O(sd)$. Further, if f_0 is not homogeneous, then $\mathcal{A}$ can be thought of as a sum of homogeneous ABPs $\{\mathcal{A}_1, \ldots, \mathcal{A}_d\}$ where $\mathcal{A}_k$ computes the k-th homogeneous component of f. Finally, if f_0 is abecedarian with respect to $\{X_i\}_{i=1}^m$ for $X_i = \{x_{i,j} \ : \ j \in [n_i]\}$, then so is each of its homogeneous components.

Therefore we assume that we are working with a homogeneous ABP, $\mathcal{A}$, of size $O(sd)$ computing a polynomial, f, of degree at least 1 that is abecedarian with respect to $\{X_i\}_{i=1}^m$ for $X_i = \{x_{i,j} \ : \ j \in [n_i]\}$. We prove the theorem by describing how to construct $\mathcal{A}'$ using $\mathcal{A}$.

For each vertex v in $\mathcal{A}$, make $O(m)$ copies in $\mathcal{A}'$, namely $\{(v, a) \ : \ 0 \leq a \leq m\}$. Further, suppose s and t are the *start* and *terminal* vertices in $\mathcal{A}$. Then we define a

[1] Every edge is labelled by a homogeneous form.

new start vertex s' in $\mathcal{A}'$ and add edges labelled with 1 from s' to each of the vertices $\{(s,a) \ : \ 1 \leq a \leq m\}$. We also define (t,m) as the terminal vertex in $\mathcal{A}'$.

Intuitively, if $g_{(u,v)}$ is the polynomial computed between u and v in $\mathcal{A}$, then the polynomial computed between (u,a) and (v,b) in $\mathcal{A}'$ is $g_{(u,v)}[a,b+1)$. We ensure this property at every vertex by adding edges in $\mathcal{A}'$ as follows.

> For any two vertices u, v in $\mathcal{A}$, suppose there is an edge between them that is labelled with $\sum_{i\in[m]} \sum_{j\in[n_i]} \gamma_{i,j} x_{i,j}$. Then, for every $a, b \in [m]$ with $a \leq b$, add an edge from (u,a) to (v,b) with label $\sum_{i=a}^{b} \left(\sum_{j\in[n_i]} \gamma_{i,j} x_{i,j} \right)$.

We also, associate the bucket index a with the gate (v,a) in $\mathcal{A}'$.

By induction, one can easily show[2] that the gates in $\mathcal{A}'$ have the claimed property. Therefore the polynomial computed by $\mathcal{A}'$ is $\sum_{i=1}^{m} f[1, m+1)$ which is indeed f. Hence $\mathcal{A}'$ is indeed an abecedarian ABP computing f. Further, every vertex v in $\mathcal{A}$, there are at most $O(m)$ vertices in $\mathcal{A}'$. Therefore, the size of $\mathcal{A}'$ is $O(msd)$. □

We now define some natural classes of abecedarian polynomials and show that the logical inclusions hold. Let $\mathsf{abcVP}_{\mathsf{nc}}$ denote the class of abecedarian polynomials that can be computed by poly-sized abecedarian circuits. Similarly let $\mathsf{abcVBP}_{\mathsf{nc}}$ and $\mathsf{abcVF}_{\mathsf{nc}}$ denote the classes of abecedarian polynomials that can be computed by poly-sized abecedarian ABPs and abecedarian formulas respectively.

Observation 5.1.3. *Let* $\mathsf{abcVP}_{\mathsf{nc}}$, $\mathsf{abcVBP}_{\mathsf{nc}}$ *and* $\mathsf{abcVF}_{\mathsf{nc}}$ *denote the classes of* abecedarian *polynomials over* n *variables that can be computed by* $\mathrm{poly}(n)$ *sized* abecedarian *circuits,* abecedarian *ABPs and* abecedarian *formulas respectively. Then,*

$$\mathsf{abcVF}_{\mathsf{nc}} \subseteq \mathsf{abcVBP}_{\mathsf{nc}} \subseteq \mathsf{abcVP}_{\mathsf{nc}}.$$

Proof. Suppose $f \in \mathsf{abcVF}_{\mathsf{nc}}$. Then f is abecedarian, and in particular $f \in \mathsf{VF}_{\mathsf{nc}}$. But we know that $\mathsf{VF}_{\mathsf{nc}} \subseteq \mathsf{VBP}_{\mathsf{nc}}$, and so $f \in \mathsf{VBP}_{\mathsf{nc}}$. By Observation 5.1.2, this implies that $f \in \mathsf{abcVBP}_{\mathsf{nc}}$.

Similarly, suppose $f \in \mathsf{abcVBP}_{\mathsf{nc}}$. Then f is abecedarian, and $f \in \mathsf{VBP}_{\mathsf{nc}}$. But $\mathsf{VBP}_{\mathsf{nc}} \subseteq \mathsf{VP}_{\mathsf{nc}}$, and so $f \in \mathsf{VP}_{\mathsf{nc}}$. By Theorem 4.1.2, this implies that $f \in \mathsf{abcVP}_{\mathsf{nc}}$. This completes the proof. □

[2] Essentially, s–t paths of $\mathcal{A}$ are simply rearranged in $\mathcal{A}'$ and therefore each s–t path in $\mathcal{A}$ appears exactly once in $\mathcal{A}'$.

Consider the following modified version of the palindrome polynomial

$$\sum_{w\in\{0,1\}^n} x_{1w_1}\cdots x_{nw_n}x_{(n+1)w_n}\cdots x_{(2n)w_1}$$

where w^R denotes the reverse of the word $w \in \{0,1\}^n$. Note that the polynomial is abecedarian with respect to the partition $\{X_i \ : \ X_i = \{x_{i0}, x_{i,1}\}\}$, has an abecedarian circuit of size $\mathrm{poly}(n)$ and by Nisan's proof cannot be computed by ABPs of size $2^{o(n)}$. Therefore, by Observation 5.1.2,

$$\mathsf{abcVBP}_{\mathsf{nc}} \subsetneq \mathsf{abcVP}_{\mathsf{nc}}$$

Finally, we observe that a degree d polynomial that is computable by an abecedarian ABP of size s is also computable by an abecedarian formula of size $O(s^{\log d})$ via the usual divide-and-conquer algorithm.

Observation 5.1.4 (Converting Abecedarian ABPs into Abecedarian Formulas)**.** *Suppose f is an* abecedarian *polynomial of degree d. If there is an* abecedarian *ABP $\mathcal{A}$ of size s computing it, then there is an* abecedarian *formula $\mathcal{F}$ computing f of size $O(s^{\log d})$.*

The main result in this chapter essentially shows that the blow-up observed above, in Observation 5.1.4, is tight. We also show that in certain settings an analogous statement to Theorem 4.1.2 or Observation 5.1.2 is true for formulas.

5.2 Our Results

We begin by formally stating our main theorem in this chapter.

Theorem 5.2.1 (Separating Abecedarian Formulas and ABPs)**.** *Define*

$$\mathrm{linked\_CHSYM}_{n,d}(\mathbf{x}) = \sum_{i_0=1}^{n}\left(\sum_{i_0\le i_1\le\ldots\le i_d\le n} x_{i_0,i_1}\cdot x_{i_1,i_2}\cdots x_{i_{d-1},i_d}\right)$$

to be the linked *complete homogeneous polynomial over n-variables of degree d.*

The polynomial $\mathrm{linked\_CHSYM}_{n,d}(\mathbf{x})$ *is* abecedarian *with respect to the partition $\{X_i \ : \ i \in [n]\}$ where $X_i = \{x_{i,j} \ : \ i \le j \le n\}$. With respect to this partition,*

1. $\mathrm{linked\_CHSYM}_{n,d}(\mathbf{x})$ *has an* abecedarian *ABP of size $O(nd)$;*

2. *for $d = \log n$, any syntactically* abecedarian *formula computing the polynomial* $\text{linked\_CHSYM}_{n/2,d}(\mathbf{x})$ *has size* $n^{\Omega(\log d)}$.

That is, there is a super-polynomial separation between syntactically abecedarian *formulas and ABPs.*

Note that in the above theorem, the partition is of size n. However, our next result shows that if the partition size is $O(\log s)$, then formulas computing abecedarian polynomials can be assumed to be abecedarian without loss of generality.

Theorem 5.2.2 (Converting Formulas into Abecedarian Formulas)**.** *Let f be an* abecedarian *polynomial with respect to a partition of size m, and $\mathcal{F}$ be a formula of size s computing f. If $m = O(\log s)$, then there is an* abecedarian *formula $\mathcal{F}'$ computing f of size* $\text{poly}(s)$.

In other words, an $n^{\omega(1)}$ lower bound against abecedarian formulas computing a polynomial that is abecedarian with respect to a partition of size $O(\log n)$, would result in a super-polynomial lower bound against general non-commutative formulas.

These statements suggest a couple of new approaches towards resolving the general VF_{nc} vs VBP_{nc} question. We first discuss these.

Connections to the General VF_{nc} vs VBP_{nc} Question

Theorem 5.2.1 gives a separation between abecedarian formulas and ABPs. On the other hand, Theorem 5.2.2 shows that if we are given a formula that computes a polynomial that is abecedarian with respect to a partition of *small* size, then we can assume that the formula is abecedarian without loss of generality. Unfortunately, the partition with respect to which our *hard polynomial* from Theorem 5.2.1 is abecedarian, is *not small* in size. Thus, the general question of whether VBP_{nc} is contained in VF_{nc} or not still remains open.

However, there are two natural questions that arise at this point.

1. Can any formula computing an abecedarian polynomial be converted to an abecedarian formula without much blow-up in size, irrespective of the size of the partition?

2. Is a separation between abecedarian formulas and ABPs witnessed by a polynomial which is abecedarian with respect to a partition that has *small* size?

Clearly, a positive answer to either of these questions would imply that $\mathsf{VBP}_{\mathrm{nc}} \neq \mathsf{VF}_{\mathrm{nc}}$. In fact, since non-commutative formulas computing polynomials of low degree can be homogenised[3] efficiently, a super-polynomial lower bound against *homogeneous* formulas for our hard polynomial (or any explicit polynomial of degree $O(\log n)$ that is efficiently computable by ABPs) would separate $\mathsf{VBP}_{\mathrm{nc}}$ and $\mathsf{VF}_{\mathrm{nc}}$.

Corollary 5.2.3. *Let* $\mathrm{linked\_CHSYM}_{n,d}(\mathbf{x})$ *be as defined in Theorem 5.2.1. An* $n^{\omega(1)}$ *lower bound against homogeneous formulas for* $\mathrm{linked\_CHSYM}_{n,\log n}(\mathbf{x})$ *would result in a super-polynomial separation between non-commutative formulas and ABPs.*

However, unfortunately, the lower bound of Limaye *et al.*[LST21a] against homogeneous non-commutative formulas does not work for polynomials of degree $O(\log n)$. Thus the general $\mathsf{VBP}_{\mathrm{nc}}$ vs $\mathsf{VF}_{\mathrm{nc}}$ question remains open.

Proof Overview

We now give a proof overview of our main theorems.

Let us first consider Theorem 5.2.2, since the proof is simpler. In order to prove the statement, we first convert the given formula $\mathcal{F}$ into an abecedarian circuit $\mathcal{C}$, and then unravel $\mathcal{C}$ to get an abecedarian formula $\mathcal{F}'$ computing the same polynomial.

The first step is fairly straightforward. The proof is along the same lines as that for homogenising circuits. The only difference is that we keep track of bucket indices of the variables on either ends of the monomials being computed at the gates, instead of their degrees.

In the second step, we convert $\mathcal{C}$ into a formula $\mathcal{F}'$. In order to do that, we need to recompute vertices every time it is reused. Thus, to give an upper bound on the size of $\mathcal{F}'$, we need to find an upper bound on the number of distinct paths from any vertex in $\mathcal{C}$ to the root. This analysis is done in a way similar to the one by Raz [Raz13] to show that formulas computing low degree polynomials can be homogenised without much blow-up in size. The requirement of the size of the partition being small also arises because of this analysis.

The only additional point that needs to be checked for the proof to go through is that similar to the commutative setting, non-commutative formulas can be depth reduced as well (Lemma 5.2.5).

[3] This follows from the proof of a similar statement in the commutative setting due to Raz [Raz13] together with the fact the non-commutative formulas can be depth-reduced to log-depth [HW15]

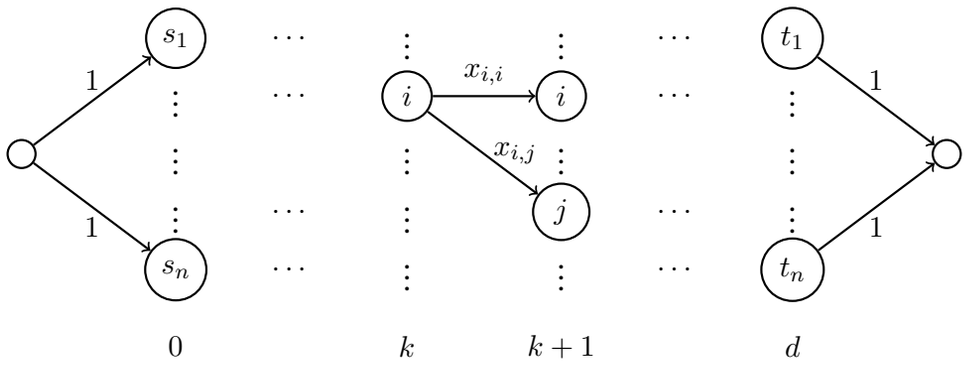


Figure 5.1.: An ABP of size $O(nd)$ computing $\text{linked\_CHSYM}_{n,d}(\mathbf{x})$.

A complete proof of Theorem 5.2.2 can be found in section 5.4.

Next we go over the proof idea of Theorem 5.2.1. A *small* abecedarian ABP computing $\text{linked\_CHSYM}_{n,d}(\mathbf{x})$ is described in Figure 5.2.

For the lower bound, we assume that a *small* syntactically abecedarian formula computing the polynomial has been given. We keep modifying this formula till we get a *small* homogeneous multilinear formula computing the *elementary symmetric polynomial* of degree $n/2$. Finally, we use the known lower bound against homogeneous multilinear formulas for this polynomial (shown by Hrubeš and Yehudayoff [HY11]), to get a contradiction.

Let us spell out the proof in some more detail.

Step 1: Assume that we are given an abecedarian formula computing the polynomial $\text{linked\_CHSYM}_{n/2,\log n}(\mathbf{x})$ of size $O(n^{\varepsilon \log\log n})$. Since the degree of the polynomial being computed is *small*, we can assume that there is in fact a *homogeneous* abecedarian formula computing $\text{linked\_CHSYM}_{n/2,\log n}(\mathbf{x})$ of size $O(n^{c\cdot\varepsilon \log\log n})$ for some constant c independent of ε.

Step 2: Using the homogeneous abecedarian formula from Step 1, we obtain a *structured* homogeneous abecedarian formula, of size $O(n^{c\cdot\varepsilon \log\log n})$, computing $\text{linked\_CHSYM}_{n/2,\log n}(\mathbf{x})$.

Step 3: We consider the complete homogeneous polynomial over n variables of degree d

$$\text{CHSYM}_{n,d}(\mathbf{x}) = \sum_{1\le i_1\le\ldots\le i_d\le n} x_{i_1}\cdots x_{i_d},$$

and show that there is a homogeneous abecedarian formula of size $\text{poly}(n)$ that computes $\text{CHSYM}_{n/2,\log n}(\mathbf{x})$.

Step 4: If the formula in Step 2 has size s and that in Step 3 has size s', then we show that there is a homogeneous abecedarian formula of size $(s \cdot s')$ computing $\mathrm{CHSYM}_{n/2,\log^2 n}(\mathbf{x})$.

Step 5: Next, we show that Step 4 can be used repeatedly at most $O(\log n / \log\log n)$ times, to obtain a homogeneous abecedarian formula computing the polynomial $\mathrm{CHSYM}_{n/2,n/2}(\mathbf{x})$, of size $O(n^{c\cdot\varepsilon\log n})$.

Step 6: Using the formula obtained in Step 5, we get a homogeneous multilinear formula computing the elementary symmetric polynomial of degree $n/2$, of size $O(n^{c\cdot\varepsilon\log n})$.

Step 7: Finally, we choose an ε that contradicts the theorem in [HY11].

Note that in order for the lower bound proof of Hrubeš and Yehudayoff to work, the degree of the polynomial needs to be $\mathrm{poly}(n)$. However in order to make the formula structured enough for it to work, we need the degree to be low so that the size does not blow up by much in the process. Therefore the crucial step in the proof is that we use the structure we get because we are in the low degree setting to amplify the degree of the polynomial being computed in a systematic way (without blowing up the size by much). This is the Step 4 in the description above.

A complete proof of Theorem 5.2.1 can be found in section 5.5.

We now elaborate a little on the first step. These statements are known to be true in the commutative setting and their proofs in the abecedarian setting are fairly similar to the ones for their commutative counterparts. We state them here nevertheless, since they might be of independent interest.

Homogenising Abecedarian Formulas computing Polynomials of Low Degree Raz [Raz13] had shown that if there is a formula computing a homogeneous polynomial of *low* degree in the commutative world, then it can be assumed without loss of generality that the formula is homogeneous. We show that this statement is true even in the non-commutative setting.

Lemma 5.2.4 (Homogenising Abecedarian Formulas computing Low Degree Polynomials)**.** *Suppose f is a non-commutative homogeneous polynomial that can be computed by a fan-in 2 formula, $\mathcal{F}$, of size s, and has degree $d = O(\log s)$. Then there is a homogeneous formula $\mathcal{F}'$ computing f, that has size $\mathrm{poly}(s)$ and whose multiplication gates have fan-in 2. Further, if $\mathcal{F}$ was* abecedarian *with respect to some partition, then $\mathcal{F}'$ is also* abecedarian *with respect to the same partition.*

The only thing that needs to be checked for Raz's proof to work in this setting is whether abecedarian formulas can be depth-reduced to log-depth. It turns out that in fact they can be.

Depth Reduction for Abecedarian Formulas Brent [Bre74] had shown that if there is a formula of size s computing a commutative polynomial f, then there is a formula of depth $O(\log s)$ and size $\mathrm{poly}(s)$ that computes the same polynomial. A similar statement was shown by Hrubeš and Wigderson [HW15] in the non-commutative setting[4]. We show that the statement continues to be true for abecedarian formulas. The proof is exactly along the same lines as the one by Brent [Bre74].

Lemma 5.2.5 (Depth Reduction of Abecedarian Formulas)**.** *If there is a fan-in 2 formula $\mathcal{F}$ of size s computing a non-commutative polynomial f, then there is a fan-in 2 formula $\mathcal{F}'$ of size $\mathrm{poly}(s)$ and depth $O(\log(s))$ computing f. Further if $\mathcal{F}$ is homogeneous, $\mathcal{F}'$ is also homogeneous. Similarly, if $\mathcal{F}$ is* abecedarian *with respect to some partition, then $\mathcal{F}'$ is also* abecedarian *with respect to the same partition.*

We first prove the structural statements mentioned above, namely Lemma 5.2.5 and Lemma 5.2.4, in the next section. Following that, in section 5.4, we prove that in certain settings, formulas computing an abecedarian polynomial can be converted into abecedarian formulas without causing much blow-up in size. Finally, we prove our main theorem that shows a super-polynomial separation between the powers of abecedarian formulas and circuits in section 5.5.

5.3 Structural Statements

In this section, we prove two structural statements in the abecedarian setting that are known to be true in the commutative setting. Apart from being crucial to our proofs, they are possibly interesting observations in their own right.

Depth Reduction for Abecedarian Formulas

Brent [Bre74] had shown that if there is a formula of size s computing a commutative polynomial f, then there is a formula of depth $O(\log s)$ and size $\mathrm{poly}(s)$ that

[4] They in fact showed it for rational functions

computes the same polynomial. This was shown to be true in the non-commutative setting as well, in more generality for rational functions, by Hrubeš *et al.* [HW15].

We show that a similar statement continues to hold for abecedarian formulas. The proof is essentially the same as the one by Brent [Bre74], just analysed carefully. We give a complete proof for the sake of completeness.

We first restate the statement.

Lemma 5.2.5 (Depth Reduction of Abecedarian Formulas)**.** *If there is a fan-in* 2 *formula* $\mathcal{F}$ *of size* s *computing a non-commutative polynomial* f*, then there is a fan-in* 2 *formula* $\mathcal{F}'$ *of size* $\mathrm{poly}(s)$ *and depth* $O(\log(s))$ *computing* f*. Further if* $\mathcal{F}$ *is homogeneous,* $\mathcal{F}'$ *is also homogeneous. Similarly, if* $\mathcal{F}$ *is* abecedarian *with respect to some partition, then* $\mathcal{F}'$ *is also* abecedarian *with respect to the same partition.*

Proof. We begin by making the following claim.

Claim 5.3.1. *Suppose* $\mathcal{F}_0$ *is a formula computing a polynomial* f_0 *and has fan-in 2. Then the there exist sub-formulas,* $L, \mathcal{F}_1, R, \mathcal{F}_2$*, of* $\mathcal{F}_0$ *such that*

- $\mathcal{F}_0' = L \cdot \mathcal{F}_1 \cdot R + \mathcal{F}_2$ *also computes* f_0*;*
- *each of* $L, \mathcal{F}_1, R, \mathcal{F}_2$ *have size at most* $(2s/3)$*;*
- *if* $\mathcal{F}_0$ *is homogeneous, then so are* $L, \mathcal{F}_1, R, \mathcal{F}_2$*;*
- *if* $\mathcal{F}_0$ *is* abecedarian *with respect to some partition,* f_{left}*,* f_1*,* f_{right}*,* f_2 *are polynomials computed by* $L, \mathcal{F}_1, R, \mathcal{F}_2$ *respectively and* $f_0 = f_0[a, b)$*, then* $f_2 = f_2[a, b)$ *and*
  - *each of* $L, \mathcal{F}_1, R, \mathcal{F}_2$ *are* abecedarian *with respect to the same partition as* $\mathcal{F}_0$
  - *when* $a = b$*,* $\quad f_{\mathsf{left}} = f_{\mathsf{left}}[a, a) \quad f_1 = f_1[a, a) \quad f_{\mathsf{right}} = f_{\mathsf{right}}[a, a)$*;*
  - *when* $a < b$*, there exist* $a \le i \le j \le b$ *such that*

$$
\begin{array}{lcccc}
a = i < j = b & \implies & f_{\mathsf{left}} = f_{\mathsf{left}}[a, i) & f_1 = f_1[i, j) & f_{\mathsf{right}} = f_{\mathsf{right}}[j, b). \\
a = i = j < b & \implies & f_{\mathsf{left}} = f_{\mathsf{left}}[a, i) & f_1 = f_1[i, j) & f_{\mathsf{right}} = f_{\mathsf{right}}[j, b). \\
a = i < j < b & \implies & f_{\mathsf{left}} = f_{\mathsf{left}}[a, i) & f_1 = f_1[i, j+1) & f_{\mathsf{right}} = f_{\mathsf{right}}[j, b). \\
a < i = j = b & \implies & f_{\mathsf{left}} = f_{\mathsf{left}}[a, i+1) & f_1 = f_1[i, j) & f_{\mathsf{right}} = f_{\mathsf{right}}[j, b). \\
a < i = j < b & \implies & f_{\mathsf{left}} = f_{\mathsf{left}}[a, i+1) & f_1 = f_1[i+1, j+1) & f_{\mathsf{right}} = f_{\mathsf{right}}[j, b). \\
a < i < j = b & \implies & f_{\mathsf{left}} = f_{\mathsf{left}}[a, i+1) & f_1 = f_1[i, j) & f_{\mathsf{right}} = f_{\mathsf{right}}[j, b). \\
a < i < j < b & \implies & f_{\mathsf{left}} = f_{\mathsf{left}}[a, i+1) & f_1 = f_1[i, j+1) & f_{\mathsf{right}} = f_{\mathsf{right}}[j, b).
\end{array}
$$

Before proving Claim 5.3.1, let us complete the proof of Lemma 5.2.5 using it.

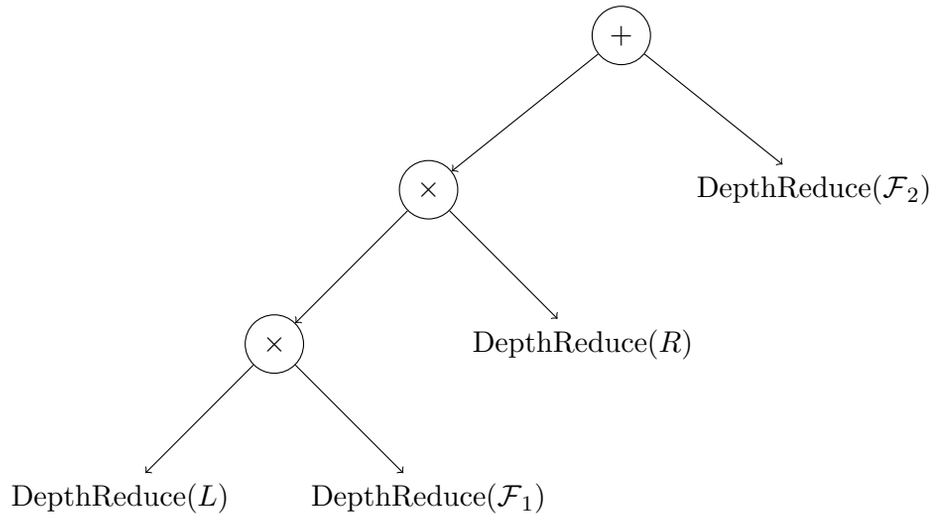
Assuming $\mathcal{F}_0$ to be the given formula $\mathcal{F}$, using the above claim, we have a formula $\mathcal{F}'_0$ computing f_0 that looks like $L \cdot \mathcal{F}_1 \cdot R + \mathcal{F}_2$ where each of $L, \mathcal{F}_1, R, \mathcal{F}_2$ have size at most $(2s/3)$. Further if $\mathcal{F}$ is homogeneous, then so are each of $L, \mathcal{F}_1, R, \mathcal{F}_2$. Hence, $\mathcal{F}'_0$ is homogeneous. On the other hand, when $\mathcal{F}_0$ is abecedarian, so are $L, \mathcal{F}_1, R, \mathcal{F}_2$. Further, note that $\mathcal{F}'_0$ is also abecedarian in this case since $f_{\mathsf{left}}, f_1, f_{\mathsf{right}}, f_2$ are of the *correct type* due to Claim 5.3.1.

In all the cases, recursively applying this technique, on each of $L, \mathcal{F}_1, R, \mathcal{F}_2$, we get

$$\mathsf{depth}(s) \leq \mathsf{depth}(2s/3) + 3 \qquad \text{and} \qquad \mathsf{size}(s) \leq 4 \cdot \mathsf{size}(2s/3) + 3.$$

Note that in the base case, when s is constant, both $\mathsf{size}(s)$ and $\mathsf{depth}(s)$ are constants. Thus, $\mathsf{depth}(s) = O(\log s)$ and $\mathsf{size}(s) = \mathrm{poly}(s)$. □

Pictorially, once we have Claim 5.3.1, we essentially do the following recursively.



We now complete the proof of Claim 5.3.1

Proof of Claim 5.3.1. From the root let us traverse $\mathcal{F}_0$ towards the leaves, always choosing the child that has a larger sub-tree under it, till we find a vertex v such that the associated sub-tree has size at most $(2s/3)$. Since $\mathcal{F}_0$ tree has fan-in 2, we also know that the size of this sub-tree must be at least $(s/3)$.

Let this sub-tree be $\mathcal{F}_1$. Additionally, in the case when $\mathcal{F}_0$ is abecedarian, let us assume that v is labelled with $[i_v, j_v)$.

Let $\mathcal{P}$ be the path from v to the root and v_{add} the addition gate on $\mathcal{P}$ which is closest to v. Also let the set of multiplication gates on $\mathcal{P}$ be $\{v_1, \ldots, v_\ell\}$ for some $\ell \in \mathbb{N}$. Assume, without loss of generality, that v_1 is closest to v and v_ℓ to the root. Further, for every $i \in [\ell]$, let L_i be sub-formula corresponding to the left child of v_i and R_i the one to its right child. Note that for every $i \in [\ell]$, exactly one of children of v_i is a vertex in $\mathcal{P}$. We can then define L and R as follows.

Step 1: Set $L = R = 1$.

Step 2: For i from 1 to ℓ,

$$L = \begin{cases} L_i \times L & \text{if the right child of } v_i \text{ is a vertex in } \mathcal{P}, \\ L & \text{otherwise.} \end{cases}$$

and

$$R = \begin{cases} R & \text{if the right child of } v_i \text{ is a vertex in } \mathcal{P}, \\ R \times R_i & \text{otherwise.} \end{cases}$$

Also define $\mathcal{F}_2$ to be the formula we get by replacing the vertex v and the sub-tree under it with 0, and then removing the redundant gates.

By construction, $\mathcal{F}_1$, L, R and $\mathcal{F}_2$ are sub-formulas of $\mathcal{F}_0$. Further, $\mathcal{F}_1$ is disjoint from L, R and $\mathcal{F}_2$. As a result, since $\mathcal{F}_1$ has size at least $(s/3)$ and at most $(2s/3)$, it must be the case that each of L, R and $\mathcal{F}_2$ have size at most $(2s/3)$.

Also, it is not hard to see that $\mathcal{F}_0' = L \cdot \mathcal{F}_1 \cdot R + \mathcal{F}_2$ computes f_0. What is left to check is that when $\mathcal{F}_0$ is homogeneous or abecedarian, then $L, \mathcal{F}_1, R, \mathcal{F}_2$ have the additional properties claimed. The one line proof of this is that each *parse-tree*[5] of $\mathcal{F}_0$ is merely restructured in the above process, without changing its value. We however go over the proof explicitly for the sake of completeness.

When $\mathcal{F}_0$ is homogeneous, since $L, \mathcal{F}_1, R, \mathcal{F}_2$ are sub-formulas, they are also homogeneous. On the other hand, suppose $\mathcal{F}_0$ is abecedarian and $f_0 = f_0[a, b)$. Recall that the vertex v was labelled by $[i_v, j_v)$. Let us set $i = i_v$ and $j = j_v$. Then, by definition, $\mathcal{F}_1$ is labelled by $[i, j)$. Hence, if f_1 is the polynomial computed at v, then $f_1 = f_1[i, j)$. Further, $\mathcal{F}_1$ is abecedarian since it is a sub-formula of $\mathcal{F}_0$ and computes an abecedarian polynomial.

Now let us focus on $\mathcal{F}_2$. Essentially $\mathcal{F}_2$ is got by removing from $\mathcal{F}_0$, v and all the multiplication gates on P between v and v_{add} along with the sub-trees under them.

[5] For a definition, see for example [LLS19]

Thus $\mathcal{F}_2$ is also abecedarian in this case, and if f_2 is the polynomial computed by it, then $f_2 = f_2[a, b)$.

Finally, note that the left indices of labels on the various vertices of $\mathcal{P}$ change only at the gates at which multiplications to L occur. Further, note that they occur in the *correct order* and are of the *correct type*. Thus, by induction, it is easy to see that the labels on L are consistent with those on the L_is when the respective multiplications happen. Therefore L is abecedarian, and $f_{\mathsf{left}} = f_{\mathsf{left}}[a, i)$. For similar reasons, R is also abecedarian and $f_{\mathsf{right}} = f_{\mathsf{right}}[j, b)$. This completes the proof. □

Homogenisation

Raz [Raz13] had shown that if there is a formula computing a homogeneous polynomial of *low* degree in the commutative world, then it can be assumed without loss of generality that the formula is homogeneous. We show that his proof also works in the abecedarian setting because of Lemma 5.2.5. A complete proof is given here for the sake of completeness.

Lemma 5.2.4 (Homogenising Abecedarian Formulas computing Low Degree Polynomials)**.** *Suppose f is a non-commutative homogeneous polynomial that can be computed by a fan-in 2 formula, $\mathcal{F}$, of size s, and has degree $d = O(\log s)$. Then there is a homogeneous formula $\mathcal{F}'$ computing f, that has size $\mathrm{poly}(s)$ and whose multiplication gates have fan-in 2. Further, if $\mathcal{F}$ was* abecedarian *with respect to some partition, then $\mathcal{F}'$ is also* abecedarian *with respect to the same partition.*

Proof. We first note that if $\mathcal{F}$ has depth r, then by Lemma 5.2.5, we can assume without loss of generality, that $r = O(\log s')$.

In order to construct a homogeneous formula computing f, we first homogenise $\mathcal{F}$ to obtain a circuit $\mathcal{C}$, and then *unravel* $\mathcal{C}$ to make it into a formula $\mathcal{F}'$.

The first step is done in the usual manner. For every gate v in $\mathcal{F}$, we have $d + 1$ gates $(v, 0)$, …, (v, d) in $\mathcal{C}$. Intuitively if f_v is the polynomial computed at v, then the polynomial computed at (v, i) is the degree i homogeneous component of f_v. These vertices are then connected as follows.

- If $v = u_1 + u_2$, then for every $i \in \{0, \ldots, d\}$, $\quad (v, i) = (u_1, i) + (u_2, i)$.
- If $v = u_1 \times u_2$, then for every $i \in \{0, \ldots, d\}$, $\quad (v, i) = \sum_{j=0}^{i} (u_1, j) \times (u_2, i - j)$.

So, we now have a homogeneous circuit $\mathcal{C}$ that computes f and has size at most $O(d^2 \cdot s')$. Also, the depth of this circuit is at most twice that of $\mathcal{F}$, and the multiplication gates have fan-in 2.

To convert $\mathcal{C}$ into a formula $\mathcal{F}'$, we have to recompute nodes whenever they have to be reused. That is, a particular vertex in $\mathcal{C}$ has to be duplicated as many times as there are paths from it to the root. Thus, to upper bound the size of $\mathcal{F}'$, we need to give a bound on the number of distinct paths from every vertex of $\mathcal{C}$ to its root.

Let us arbitrarily choose a vertex (v, i) in $\mathcal{C}$, and consider a path from it to the root. Suppose the path is $(v, i) = (v_1, i_1) \to \cdots \to (v_\ell, i_\ell) = (\text{root}, d)$ where ℓ is at most the depth of $\mathcal{C}$. By construction, it must be that $v = v_1 \to \cdots \to v_\ell = \text{root}$ is the unique path from v to the root in $\mathcal{F}$. Therefore it is only the number of possible *second co-ordinates* in the path description that we need to bound.

Now it must be the case that $i = i_1 \leq \cdots \leq i_\ell = d$. Hence, if we define $\delta_j = i_{j+1} - i_j$ for $j \in [\ell - 1]$, then the δ_js are non-negative integers such that $\delta_1 + \cdots + \delta_{\ell-1} = (d - i)$. Thus, the number of choices we have for $(i_2, \ldots, i_\ell)$ such that $i = i_1 \leq \cdots \leq i_\ell = d$, is the same as the number of choices we have for $(\delta_1, \ldots, \delta_{\ell-1})$ such that $\delta_1 + \cdots + \delta_{\ell-1} = (d - i) \leq d$. This is at most $\binom{\ell+d}{\ell}$.

Note that in this process the fan-in of the gates have not changed, and hence the multiplication gates in $\mathcal{F}'$ continue to have fan-in 2. Further, we know that the $\mathcal{C}$ has depth $2r$ and hence $\ell \leq 2r$. Therefore, the number of paths from (v, i) to the root is at most $\binom{2r+d}{2r}$. Hence, if $\mathcal{F}'$ is the formula obtained by unravelling $\mathcal{C}$, then $\text{size}(\mathcal{F}') \leq s' \cdot d^2 \cdot \binom{2r+d}{d}$. Here $r = O(\log(s'))$, and $s \leq s'$ implying that $d = O(\log(s)) = O(\log(s'))$. Thus, $\text{size}(\mathcal{F}') \leq \text{poly}(s')$.

Finally, assume that $\mathcal{F}$ is abecedarian. Then every vertex v is labelled with a tuple of bucket indices, say (a_v, b_v). In that case, we add the label (a_v, b_v) to the gates $\{(v, i)\}_{i=0}^{d}$ in $\mathcal{C}$ and continue with the proof as is. Note that the final formula that we get, $\mathcal{F}'$, is abecedarian and all the other properties that were true in the general case, continue to be true. □

As a corollary, we now have that proving a lower bound against homogeneous non-commutative formulas would be enough to separate VBP_{nc} and VF_{nc}.

Corollary 5.2.3. *Let* $\text{linked\_CHSYM}_{n,d}(\mathbf{x})$ *be as defined in Theorem 5.2.1. An* $n^{\omega(1)}$ *lower bound against homogeneous formulas for* $\text{linked\_CHSYM}_{n,\log n}(\mathbf{x})$ *would result in a super-polynomial separation between non-commutative formulas and ABPs.*

Proof. By Theorem 5.2.1, we know that $\text{linked\_CHSYM}_{n,\log n}(\mathbf{x})$ can be computed by ABPs of size $\text{poly}(n)$. Further, the degree of the polynomial is $O(\log n)$. Thus, by Lemma 5.2.4, if there is a formula computing $\text{linked\_CHSYM}_{n,\log n}(\mathbf{x})$ of size s, then there is a homogeneous formula computing it of size $\text{poly}(s)$. This immediately implies the given statement. □

We now move on to proving our main theorems. First we show that even in the case of formulas, in certain settings, we can convert formulas computing abecedarian polynomials into abecedarian formulas with only a polynomial blow-up in size.

5.4 Converting Formulas into Abecedarian Formulas

We have already seen that without loss of generality, circuits and ABPs computing abecedarian polynomials can be assumed to be abecedarian. For formulas however, we can prove such a statement only when the polynomial is abecedarian with respect to a bucketing system of *small size*. The proof is very similar to that of Lemma 5.2.4.

Theorem 5.2.2 (Converting Formulas into Abecedarian Formulas)**.** *Let f be an* abecedarian *polynomial with respect to a partition of size m, and $\mathcal{F}$ be a formula of size s computing f. If $m = O(\log s)$, then there is an* abecedarian *formula $\mathcal{F}'$ computing f of size* $\text{poly}(s)$.

Proof. Let us assume additionally that $\mathcal{F}$ has depth r. Now Lemma 5.2.5 implies that $r = \log(s)$ without loss of generality. By Theorem 4.1.2, there is an abecedarian circuit $\mathcal{C}$ that computes f and has size at most $s' = O(s \cdot m^3)$. Further its proof implies that the depth of $\mathcal{C}$ is at most $2r$.

To convert $\mathcal{C}$ into an abecedarian formula $\mathcal{F}'$, we have to recompute a node each time it has to be reused. That is, a particular vertex in $\mathcal{C}$ has to be duplicated as many times as there are paths from the vertex to the root. Thus to upper bound the size of $\mathcal{F}'$, we need to give an upper bound on the number of distinct paths from every vertex in $\mathcal{C}$ to its root.

Let us arbitrarily choose a vertex $(v, [a, b))$ in $\mathcal{C}$, and consider the path from it to the root. Suppose the path is $(v, [a, b)) = (v_1, [a_1, b_1)) \to \cdots \to (v_\ell, [a_\ell, b_\ell)) = (\text{root}, [i, m + 1))$ for some ℓ that is at most the depth of $\mathcal{C}$. Again, as in the proof of Lemma 5.2.4, we only need to bound the different possibilities in the second

co-ordinate. This is because, by construction, $v = v_1 \to \cdots \to v_\ell = \text{root}$ must be the unique path from v to root in $\mathcal{F}$.

Now note that it must be the case that

$$i \le a_\ell \le \cdots \le a_1 \le a \le b \le b_1 \le b_\ell \le m+1.$$

Let us define $\delta_j = a_j - a_{j+1}$ and $\delta'_j = b_{j+1} - b_j$ for $j \in [\ell - 1]$. Then, the number of choices we have for $(a_1, \ldots, a_\ell)$ and $(b_1, \ldots, b_\ell)$ such that

$$i = a_\ell \le \cdots a_1 = a \le b = b_1 \le \cdots \le b_\ell = m+1$$

is the same as the number of choices we have for $(\delta_1, \ldots, \delta_{\ell-1}, \delta'_1, \ldots, \delta'_{\ell-1})$ such that

$$\delta_1 + \cdots + \delta_{\ell-1} + \delta'_1 + \cdots + \delta'_{\ell-1} = (m + 1 - (b-a) - i) \le m.$$

This is clearly at most $\binom{2\ell+m}{m}$.

Further, we know that the $\mathcal{C}$ has depth $2r$ and hence $\ell \le 2r$. Therefore, the number of paths from (v, i) to the root is at most $\binom{4r+m}{m}$. Hence if $\mathcal{F}'$ is the formula obtained by unravelling $\mathcal{C}$, then $\text{size}(\mathcal{F}') \le s' \cdot m^2 \cdot \binom{4r+m}{m}$. Here $s' = O(m^3 \cdot s)$, $r = O(\log(s))$ and $m = O(\log(s))$. Thus, $\text{size}(\mathcal{F}') \le \text{poly}(s)$. □

Finally, in the next section, we prove our main theorem – a super-polynomial separation between the powers of abecedarian formulas and ABPs.

5.5 Separating Abecedarian Formulas and ABPs

Before proceeding to the proof, let us go over some observations that we will need.

Some Simple Observations

The two main polynomials we will be working with are $\text{linked\_CHSYM}_{n,d}(\mathbf{x})$ and $\text{CHSYM}_{n,d}(\mathbf{x})$. Let us recall their definitions.

$$\text{linked\_CHSYM}_{n,d}(\mathbf{x}) = \sum_{i_0=1}^{n} \left(\sum_{i_0 \le i_1 \le \ldots \le i_d \le n} x_{i_0,i_1} \cdot x_{i_1,i_2} \cdots x_{i_{d-1},i_d} \right),$$

is abecedarian with respect to the bucketing system $\{X_1, \ldots, X_n\}$ where $X_i = \{x_{i,j} \ : \ j \in [n]\}$, and

$$\mathrm{CHSYM}_{n,d}(\mathbf{x}) = \sum_{1 \le i_1 \le \ldots \le i_d \le n} x_{i_1} \cdots x_{i_d}.$$

is abecedarian with respect to the bucketing system $\{X_i \ : \ X_i = \{x_i\}\}$.

We begin with the notion of a *linked* abecedarian formula computing the polynomial $\mathrm{linked\_CHSYM}_{n,d}(\mathbf{x})$.

Definition 5.5.1. *An* abecedarian *formula computing* $\mathrm{linked\_CHSYM}_{n,d}(\mathbf{x})$ *is said to be* linked *if at every gate, all the monomials occurring in the polynomial computed at that gate have the following property.*

x_{ij} appears right before $x_{i'j'}$ in the monomial $\implies j = i'$. $\Diamond$

The first observation shows that any abecedarian formula computing the polynomial $\mathrm{linked\_CHSYM}_{n,d}(\mathbf{x})$ can be assumed to be *linked* without loss of generality.

Observation 5.5.2. *Let $\mathcal{F}$ be a homogeneous* abecedarian *formula computing the polynomial* $\mathrm{linked\_CHSYM}_{n,d}(\mathbf{x})$ *of size s, and let the multiplication gates of $\mathcal{F}$ have fan-in 2. Then there is a homogeneous* linked abecedarian *formula $\mathcal{F}'$ computing the same polynomial of size $O(s)$.*

Proof. For any leaf ℓ in $\mathcal{F}$ labelled by a variable, say $x_{i,j}$, suppose $\mathcal{P}$ is the path from ℓ to the root. Consider the set of multiplication gates on $\mathcal{P}$ whose left child is part of $\mathcal{P}$, and let v be the one that is closest to ℓ. Since $\mathcal{F}$ is abecedarian, the right child of v must be associated with a set, say $[a, b)$. If $j \neq a$, we set the label of ℓ to zero; otherwise we let it be $x_{i,j}$.

Note that this operation does not kill any *valid* monomial. Let $\mathcal{F}'$ be the formula we get by performing the above operation on every leaf of $\mathcal{F}$ that is labelled by a variable. $\mathcal{F}'$ is clearly homogeneous and abecedarian. We show that $\mathcal{F}'$ is also *linked*.

Suppose that is not the case. Then there must be a *problematic* vertex in $\mathcal{F}'$. Let v be such a vertex of minimal height. That is, there is a monomial in the polynomial computed at v in which, say, $x_{i,j}$ appears right before $x_{i',j'}$ but $j \neq i'$. Further, the sub-formulas corresponding to the children of v are linked. Note that v must be a multiplication gate; not a leaf or an addition gate.

Let f_{left} and f_{right} be the polynomials computed at the left and right children of v respectively. Also, let $[a, b)$ be the set associated with the right child of v. Then, it must be the case that the first variable in any monomial in f_{right} looks like $x_{a,j'}$ for some j'. Further, there must be a monomial in f_{left} in which the last variable looks like $x_{i,j}$ for $j \neq a$.

Look at the leaf corresponding to this variable. Let this leaf be ℓ and let $\mathcal{P}$ be the path from ℓ to the root. Since $x_{i,j}$ is the right most variable in f_{left}, it must be the case that v is the multiplication gate that is closest to ℓ, whose left child is on $\mathcal{P}$. But then, we should have set $x_{i,j}$ to zero since $j \neq a$. Hence, such a monomial cannot appear in f_{left}.

This shows that $\mathcal{F}'$ is indeed a homogeneous *linked* abecedarian formula of size at most that of $\mathcal{F}$ that computes $\mathrm{linked\_CHSYM}_{n,d}(\mathbf{x})$. □

The next observation shows that there is a poly-sized homogeneous abecedarian formula that computes $\mathrm{CHSYM}_{n,\log n}(\mathbf{x})$.

Observation 5.5.3. *The polynomial* $\mathrm{CHSYM}_{n/2,\log n}(\mathbf{x})$ *can be computed by a homogeneous* abecedarian *formula of size* $\mathrm{poly}(n)$.

Proof. Consider the following polynomial over variables $\{t, x_1, \ldots, x_n\}$, where we think of t as a commuting variable and $x_1, \ldots, x_n$ as non-commuting variables.

$$f_{n,d}(\mathbf{x}) = \prod_{i=1}^{n} \left(1 + \sum_{j=1}^{d} t^j \cdot x_i^j\right)$$

Note that the coefficient of t^d in $f_{n,d}(\mathbf{x})$ is exactly $\mathrm{CHSYM}_{n,d}(\mathbf{x})$. Further, it is not hard to see that $f_{n/2,\log n}(\mathbf{x})$ is abecedarian in terms of $\mathbf{x}$ with respect to the bucketing system $\{X_i \ : \ X_i = \{x_i\}\}$, and that the given expression results in an abecedarian formula of size $O(n(\log n)^2)$.

Since t is a commuting variable, we can use the usual interpolation techniques [BC92], to get an abecedarian formula of size $O(n \log n \cdot n(\log n)^2) = O(n^2(\log n)^3) = \mathrm{poly}(n)$ that computes $\mathrm{CHSYM}_{n/2,\log n}(\mathbf{x})$. Since the degree of $\mathrm{CHSYM}_{n/2,\log n}(\mathbf{x})$ is $O(\log n)$, by Lemma 5.2.4, there is a homogeneous abecedarian formula computing $\mathrm{CHSYM}_{n/2,\log n}(\mathbf{x})$ of size $\mathrm{poly}(n)$. □

Another simple observation is that if we are given a homogeneous abecedarian formula for an abecedarian polynomial, then we almost immediately have one for its various sub-polynomials.

Observation 5.5.4. *Suppose there is a homogeneous* abecedarian *formula $\mathcal{F}$ computing a polynomial f that is* abecedarian *with respect to a bucketing system of size m. Then, for any $a, b \in [m+1]$, there is a homogeneous* abecedarian *formula $\mathcal{F}_{a,b}$ of size s that computes $f[a, b)$.*

Proof. Recall that if $\mathcal{F}$ is a homogeneous abecedarian formula computing f, then $\mathcal{F}$ is in fact a set of formulas $\{\mathcal{F}_i : \mathcal{F}_i \text{ computes } f[i, m+1)\}$. Consider the formula $\mathcal{F}_a$ and set all variables that belong to buckets $\{X_b, \ldots, X_m\}$ to zero in $\mathcal{F}_a$. This operation clearly kills exactly the monomials in $f[a, m+1)$ that are not in $f[a, b)$. Thus if we call this new formula $\mathcal{F}_{a,b}$, then $\mathcal{F}_{a,b}$ is homogeneous, abecedarian and computes $f[a, b)$. □

The next observation is extremely crucial, since it allows us to *amplify the degree* of $\mathrm{CHSYM}_{n,d}(\mathbf{x})$.

Lemma 5.5.5. *Suppose there is a homogeneous* abecedarian *formula computing $\mathrm{CHSYM}_{n,d}(\mathbf{x})$ of size s, and a homogeneous* linked abecedarian *formula computing $\mathrm{linked\_CHSYM}_{n,d'}(\mathbf{x})$ of size s'. Then, there is a homogeneous* abecedarian *formula computing $\mathrm{CHSYM}_{n,(d\cdot d')}(\mathbf{x})$ of size $(s \cdot s')$.*

Proof. Let $\mathcal{F}$ be the homogeneous abecedarian formula computing the polynomial $\mathrm{CHSYM}_{n,d}(\mathbf{x})$ of size s, and $\mathcal{F}'$ be the homogeneous *linked* abecedarian formula computing $\mathrm{linked\_CHSYM}_{n,d'}(\mathbf{x})$ of size s'.

We think of the variable $x_{a,b}$ in $\mathrm{linked\_CHSYM}_{n,d'}(\mathbf{x})$ as a placeholder for the sub-polynomial $\mathrm{CHSYM}_{n,d}[a, b+1)(\mathbf{x})$[6] of $\mathrm{CHSYM}_{n,d}(\mathbf{x})$. Note that there is a bijection between monomials in $\mathrm{CHSYM}_{n,(d\cdot d')}(\mathbf{x})$ and those in the polynomial we get by substituting $x_{a,b}$ in $\mathrm{linked\_CHSYM}_{n,d'}(\mathbf{x})$ with $\mathrm{CHSYM}_{n,d}[a, b+1)(\mathbf{x})$.

By Observation 5.5.4, there is homogeneous abecedarian formula $\mathcal{F}_{a,b}$, of size $O(s)$ computing $\mathrm{CHSYM}_{n,d}[a, b+1)(\mathbf{x})$ for every $a, b \in [n+1]$. Thus, if we replace every leaf of $\mathcal{F}'$ labelled by $x_{a,b}$ with $\mathcal{F}_{a,b}$, then the resulting formula is a homogeneous abecedarian formula computing $\mathrm{CHSYM}_{n,(d\cdot d')}(\mathbf{x})$ of size $(s \cdot s')$. □

Finally, we observe that if we are given a homogeneous abecedarian formula computing the polynomial $\mathrm{CHSYM}_{(n-d+1),d}(\mathbf{x})$, then we get a homogeneous multilinear formula computing the non-commutative version of $\mathrm{ESYM}_{n,d}(\mathbf{x})$.

[6] Sum of monomials in $\mathrm{CHSYM}_{n,d}(\mathbf{x})$ whose first variable is a and last variable is one of $\{x_a, \ldots, x_b\}$.

Observation 5.5.6. *Consider the* elementary symmetric polynomial

$$\mathrm{ESYM}_{n,d}(\mathbf{x}) = \sum_{1 \le i_1 < \ldots < i_d \le n} x_{i_1} \cdots x_{i_d}.$$

If there is a homogeneous abecedarian *formula computing* $\mathrm{CHSYM}_{(n-d+1),d}(\mathbf{x})$ *of size* s*, then there is a homogeneous multilinear formula computing* $\mathrm{ESYM}_{n,d}(\mathbf{x})$ *of size* s*.*

Proof. Suppose $\mathcal{F}$ is a homogeneous abecedarian formula computing the polynomial $\mathrm{CHSYM}_{(n-d+1),d}(\mathbf{x})$ of size s. Since $\mathcal{F}$ is homogeneous, every leaf labelled by a variable can be associated with a position index. If a leaf labelled x_i has position k associated with it, then replace the label of that leaf with x_{i+k-1}.

Call this formula $\mathcal{F}'$. Clearly $\mathcal{F}'$ is a homogeneous formula of size s computing $\mathrm{ESYM}_{n,d}(\mathbf{x})$. Further note that since $\mathcal{F}$ was abecedarian, $\mathcal{F}'$ is multilinear. □

Proof of the Separation

We now prove Theorem 5.2.1. Let us first recall the statement.

Theorem 5.2.1 (Separating Abecedarian Formulas and ABPs)**.** *Define*

$$\mathrm{linked\_CHSYM}_{n,d}(\mathbf{x}) = \sum_{i_0=1}^{n} \left(\sum_{i_0 \le i_1 \le \ldots \le i_d \le n} x_{i_0,i_1} \cdot x_{i_1,i_2} \cdots x_{i_{d-1},i_d} \right)$$

to be the linked *complete homogeneous polynomial over* n*-variables of degree* d*.*

The polynomial $\mathrm{linked\_CHSYM}_{n,d}(\mathbf{x})$ *is* abecedarian *with respect to the partition* $\{X_i \,:\, i \in [n]\}$ *where* $X_i = \{x_{i,j} \,:\, i \le j \le n\}$*. With respect to this partition,*

1. $\mathrm{linked\_CHSYM}_{n,d}(\mathbf{x})$ *has an* abecedarian *ABP of size* $O(nd)$*;*
2. *for* $d = \log n$*, any syntactically* abecedarian *formula computing the polynomial* $\mathrm{linked\_CHSYM}_{n/2,d}(\mathbf{x})$ *has size* $n^{\Omega(\log d)}$*.*

That is, there is a super-polynomial separation between syntactically abecedarian *formulas and ABPs.*

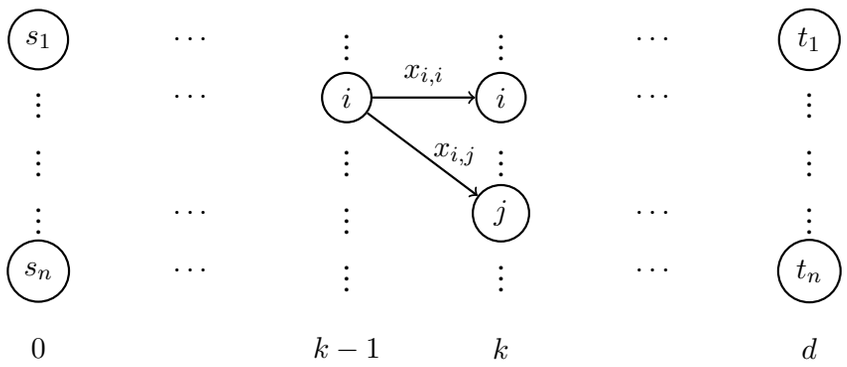
That $\mathrm{linked\_CHSYM}_{n,d}(\mathbf{x})$ has a *small* abecedarian ABP is not very hard to see. For the lower bound, we assume that we have been given an abecedarian formula $\mathcal{F}$, computing the polynomial $\mathrm{linked\_CHSYM}_{n,\log n}(\mathbf{x})$, of size $\mathrm{poly}(n)$. We then keep making changes to this formula till we get a homogeneous multilinear formula

computing $\mathrm{ESYM}_{n,n/2}(\mathbf{x})$ of size $\mathrm{poly}(n)$. Finally, we use the following theorem of Hrubeš and Yehudayoff [HY11] to get a contradiction.

Theorem 5.5.7 (Theorem 1, [HY11]). *Any homogeneous multilinear formula that computes* $\mathrm{ESYM}_{n,d}(\mathbf{x})$*, for* $d \leq n/2$*, must have size* $n \times d^{\Omega(\log d)}$.

Let us now complete the proof of our main theorem.

Proof of Theorem 5.2.1. An abecedarian ABP of size $O(nd)$ computing the polynomial $\mathrm{linked\_CHSYM}_{n,d}(\mathbf{x})$ is the following.



The ABP has $d+1$ layers, labelled 0 through d, each with n nodes. Between any consecutive layers $k-1$ and k, where $1 \leq k \leq d$, there is an edge from the i-th node in layer $k-1$ to the j-th node in layer k layer if $i \leq j$. The label on this edge is $x_{i,j}$. All the nodes in the first layer are start nodes, and all the ones in the last layer are terminal nodes.

It is easy to check, by induction, that the polynomial computed between s_a and the b-th vertex in layer k computes $\mathrm{CHSYM}_{n,k}[a, b+1)(\mathbf{x})$. Thus the polynomial computed by the abecedarian ABP constructed above is indeed $\mathrm{CHSYM}_{n,d}(\mathbf{x})$, and its size is clearly $O(nd)$.

Let us now move on to proving the lower bound against abecedarian formulas. We show that there is a fixed constant ε_0 such that any abecedarian formula computing $\mathrm{linked\_CHSYM}_{n/2,\log n}(\mathbf{x})$ must have size atleast $\Omega(n^{\varepsilon_0 \log\log n})$. Suppose this is not the case. Then for every $\varepsilon > 0$, there is an abecedarian formula $\mathcal{F}'(\varepsilon)$ computing $\mathrm{linked\_CHSYM}_{n/2,\log n}(\mathbf{x})$ of size $O(n^{\varepsilon \log\log n})$.

Without loss of generality, we can assume that $\mathcal{F}'(\varepsilon)$ has fan-in 2. Further, by Lemma 5.2.5, we can reduce the depth of $\mathcal{F}'(\varepsilon)$ to $\log$-depth. That is, we get an abecedarian formula $\mathcal{F}'_1(\varepsilon)$ computing $\mathrm{linked\_CHSYM}_{n/2,\log n}(\mathbf{x})$ of depth at most

$O(\varepsilon \log n \log\log n)$ and size $O(n^{c_1 \varepsilon \log\log n})$. Here c_1 is a fixed constant independent of ε.

Next, since the degree of the polynomial being computed is *small*, Lemma 5.2.4 implies that $\mathcal{F}'_1(\varepsilon)$ can in fact be homogenised without much blow-up in size. In other words, there is a homogeneous abecedarian formula computing the polynomial $\text{linked\_CHSYM}_{n/2,\log n}(\mathbf{x})$ of size $O(n^{c_1 c_2 \varepsilon \log\log n})$, where c_2 is again a fixed constant independent of ε. Let this formula be $\mathcal{F}'_2(\varepsilon)$.

By Observation 5.5.2, we can use $\mathcal{F}'_2(\varepsilon)$ to get a homogeneous linked abecedarian formula $\mathcal{F}'_3(\varepsilon)$ of size $O(n^{c_1 c_2 \varepsilon \log\log n})$ that computes the same polynomial. Further, because of Observation 5.5.3, we know that there is a homogeneous abecedarian formula, say $\mathcal{F}$, of size $\text{poly}(n) = O(n^{c_1 c_2 \varepsilon \log\log n})$ that computes $\text{CHSYM}_{n/2,\log n}(\mathbf{x})$.

With $\mathcal{F}$ and $\mathcal{F}'_3(\varepsilon)$ in hand, we get a homogeneous abecedarian formula computing $\text{CHSYM}_{n/2,\log^2 n}(\mathbf{x})$ because of Lemma 5.5.5. To get such a formula for $\text{CHSYM}_{n/2,n/2}(\mathbf{x})$, we need to use Lemma 5.5.5 at most k times where

$$(\log n)^k = \frac{n}{2} \implies k = O\left(\frac{\log n}{\log\log n}\right).$$

Thus, using Lemma 5.5.5 repeatedly at most $O(\log n/\log\log n)$ times, we get that there is a homogeneous abecedarian formula, $\mathcal{F}(\varepsilon)$, computing $\text{CHSYM}_{n/2,n/2}(\mathbf{x})$ of size

$$O(n^{(c_1 c_2 \varepsilon \log\log n)\cdot(\log n/\log\log n)}) = O(n^{(c_1 c_2 \varepsilon \log n)}).$$

By Observation 5.5.6, we know that $\mathcal{F}(\varepsilon)$ can be used to get a homogeneous multilinear formula, $\mathcal{F}_1(\varepsilon)$, computing $\text{ESYM}_{n-1,n/2}(\mathbf{x})$ of size $O(n^{(c_1 c_2 \varepsilon \log n)})$. Finally, Theorem 5.5.7 tells us that there is a constant δ such that any homogeneous multilinear formula computing $\text{ESYM}_{n-1,n/2}(\mathbf{x})$ must have size at least $n^{\delta \cdot \log n}$. For $\varepsilon = \delta/2c_1c_2$, this contradicts the existence of $\mathcal{F}_1(\varepsilon)$ and hence $\mathcal{F}'(\varepsilon)$.

Thus, it must be the case that any abecedarian formula computing the polynomial $\text{linked\_CHSYM}_{n/2,\log n}(\mathbf{x})$ has size at least $n^{\Omega(\log\log n)}$. This completes the proof. □

Our main result in this chapter was a tight super-polynomial separation between abecedarian formulas and algebraic branching programs. In a recent breakthrough work, Limaye, Srinivasan and Tavenas [LST21a] have shown a super-polynomial separation between homogeneous formulas and ABPs in the non-commutative setting using very different techniques. It would be very interesting to see if ideas from

their work and this work can be combined to give a solution to Nisan's question of whether there is a super-polynomial gap between the powers of formulas and ABPs in the non-commutative setting.

Part II

Algebraic Independence and Faithful Homomorphisms

Testing Algebraic Independence

6

Till now we have been focussing on questions that try to classify multivariate polynomials with respect to how hard it is to compute them for various algebraic models of computation. In this part, we look at some algorithmic tasks related to polynomials. The first task we study involves understanding certain relationships between polynomials — the concept of *algebraic independence*.

Recall that a set of polynomials $\{f_1, \ldots, f_m\} \subseteq \mathbb{F}[\mathbf{x}]$ is said to be *algebraically dependent* if and only if there is some nonzero polynomial combination of $\{f_1, \ldots, f_m\}$ that is zero. Such a nonzero polynomial $A(z_1, \ldots, z_m) \in \mathbb{F}[\mathbf{z}]$, if one exist, for which $A(f_1, \ldots, f_m) = 0$ is called an *annihilating polynomial* for the set $\{f_1, \ldots, f_m\}$.

For instance, if $f_1 = x$, $f_2 = y$ and $f_3 = x^2 + y^2$, then $A = z_1^2 + z_2^2 - z_3$ is an annihilator. Note that the underlying field is very important. For example, the polynomials $x + y$ and $x^p + y^p$ are algebraically dependent over $\mathbb{F}_p$, but algebraically independent over a characteristic zero field.

6.1 Algebraic Rank

As mentioned earlier, algebraically independent subsets of a given set of polynomials $\mathbf{f} = \{f_1, \ldots, f_m\}$ form a *matroid* (see [Oxl92]). Hence, the size of the maximum algebraically independent subset of $\mathbf{f}$ is well-defined and is called the *algebraic rank* or *transcendence degree* of $\mathbf{f}$. We denote it by $\mathsf{algrank}(\mathbf{f}) = \mathsf{algrank}(f_1, \ldots, f_m)$.

Several computational questions arise from the above definition. For instance, given a set of polynomials $\mathbf{f} = \{f_1, \ldots, f_m\}$, each f_i given as a sum of monomials, can we compute $\mathsf{algrank}(\mathbf{f})$ efficiently? What if the f_i's are provided as algebraic circuits?

Furthermore, in instances when $\mathsf{algrank}(\mathbf{f}) = m - 1$, the smallest degree annihilating polynomial is unique ([Kay09]). There could be various questions about the minimal degree annihilator in this case. For instance, can we compute it efficiently? Kayal [Kay09] showed that even checking if the constant term of the annihilator is zero is NP-hard, and evaluating the annihilator at a given point is #P-hard.

In fact, recently Guo, Saxena, Sinhababu [GSS19] showed that even in the general case, checking if the constant term of every annihilator is zero is NP-hard. This effectively rules out any attempt to compute the algebraic rank via properties of the annihilating polynomials.

Despite this, over fields of characteristic zero, algebraic rank has an alternate characterisation via the Jacobian criterion. Jacobi [Jac41] showed that the algebraic rank of a set of polynomials $\mathbf{f}(\subseteq \mathbb{F}[\mathbf{x}])$ is given by the linear rank (over the rational function field $\mathbb{F}(\mathbf{x})$) of the Jacobian of these polynomials.

6.2 The Jacobian Criterion

Let us now formally state the Jacobian Criterion.

Definition 6.2.1 (Jacobian Matrix)**.** *For $f_1, \ldots, f_m \in \mathbb{F}[x_1, \ldots, x_n]$, the Jacobian matrix is defined as*

$$\mathbf{J}_{\mathbf{x}}(\mathbf{f}) = \begin{bmatrix} \partial_{x_1}(f_1) & \partial_{x_1}(f_2) & \ldots & \partial_{x_1}(f_m) \\ \partial_{x_2}(f_1) & \partial_{x_2}(f_2) & \ldots & \partial_{x_2}(f_m) \\ \vdots & \vdots & \ddots & \vdots \\ \partial_{x_n}(f_1) & \partial_{x_n}(f_2) & \ldots & \partial_{x_n}(f_m) \end{bmatrix}$$

◊

With this definition in mind, the Jacobian criterion [Jac41] can now be stated as follows.

Theorem 6.2.2 (Jacobian Criterion)**.** *If $\mathbb{F}$ is a field of characteristic zero, then the set of polynomials $f_1, \ldots, f_m \in \mathbb{F}[\mathbf{x}]$ are algebraically independent if and only if $\mathbf{J}_{\mathbf{x}}(\mathbf{f})$ has full rank over the rational function field $\mathbb{F}(\mathbf{x})$.* □

This immediately yields a randomized polynomial time algorithm to compute the algebraic rank of a given set of polynomials by computing the rank of the Jacobian evaluated at a random point [Ore22; Sch80; Zip79; DL78].

However, this criterion is not true over fields that have finite characteristic. A standard example to exhibit the failure of the Jacobian criterion over fields of finite characteristic, is $\{x^{p-1}y, y^{p-1}x\}$ – these polynomials are algebraically independent over $\mathbb{F}_p$ but the Jacobian is *not* full-rank over $\mathbb{F}_p$.

Pandey, Saxena and Sinhababu [PSS18] characterised the extent of failure of the Jacobian criterion for $\{f_1, \ldots, f_m\}$ by a notion called the *inseparable degree* associated with this set (see Definition 6.3.4). Further, they presented a Jacobian-like criterion to capture algebraic independence.

6.3 The PSS Criterion

The exact characterisation given by Pandey *et al.* [PSS18] is slightly involved and requires some field theoretic concepts, which we present first.

Some Field Theoretic Preliminaries

Definition 6.3.1. *A polynomial is said to be separable if it does not have repeated roots in a field where it factorises completely.* ◊

Over characteristic zero fields, every irreducible univariate polynomial is separable since it cannot have a common root with its derivative. However, this is not the case over fields of finite characteristic as derivatives of non-trivial polynomials could become zero. This adds some subtlety in field extensions over finite characteristic.

Some facts about field extensions that we will need are mentioned below. These may be found in any standard text for field theory (see, for example, [Isa94]).

1. An extension $\mathbb{K}/\mathbb{F}$ is said to be algebraic if every element in $\mathbb{K}$ is the root of some polynomial over $\mathbb{F}$. Otherwise, it is transcendental.

2. For a transcendental extension $\mathbb{K}/\mathbb{F}$, a transcendence basis is a maximal subset of $\mathbb{K}$ that is algebraically independent over $\mathbb{F}$. An extension $\mathbb{K}/\mathbb{F}$ is purely transcendental if there is a transcendence base $S \subseteq \mathbb{F}$ such that $\mathbb{K} = \mathbb{F}(S)$.

3. An algebraic extension $\mathbb{K}/\mathbb{F}$ is said to be separable if the minimal polynomial of every element in $\mathbb{K}$ is separable.

   An example of an algebraic extension that is *not* separable is $\mathbb{F}_p(x)/\mathbb{F}_p(x^p)$. The minimal polynomial $\mu(z)$ for x over $\mathbb{F}_p(x^p)$ is $z^p - x^p$, which is not separable.

   Also, if $\mathbb{K} = \mathbb{F}(\alpha_1, \ldots, \alpha_n)$ is an algebraic extension of $\mathbb{F}$, then $\mathbb{K}/\mathbb{F}$ is separable if and only if the minimal polynomials of α_i over $\mathbb{F}$ is separable for each i.

For an algebraic extension $\mathbb{K}/\mathbb{F}$ over characteristic p, the *separable closure* of $\mathbb{F}$ in $\mathbb{K}$, denoted by $\mathrm{Sep}(\mathbb{K}/\mathbb{F})$, is defined as

$$\mathrm{Sep}(\mathbb{K}/\mathbb{F}) = \{\alpha \in \mathbb{K} \ : \ \alpha \text{ is separable over } \mathbb{F}\}\,.$$

For every element α in $\mathbb{K} \setminus \mathrm{Sep}(\mathbb{K}/\mathbb{F})$, we would have that $\alpha^{p^i} \in \mathrm{Sep}(\mathbb{K}/\mathbb{F})$ for some positive integer i. Thus, the extension $\mathbb{K}/\mathbb{F}$ splits into two extensions $\mathbb{K} \geq \mathrm{Sep}(\mathbb{K}/\mathbb{F}) \geq \mathbb{F}$ where the latter is a *separable algebraic* extension and the former is a *purely inseparable* algebraic extension.

Definition 6.3.2 (Inseparable degree of algebraic extensions)**.** *For an algebraic extension $\mathbb{K}/\mathbb{F}$ of characteristic p, the* inseparable degree *of the extension, denoted by* $\mathsf{insep\text{-}deg}(\mathbb{K}/\mathbb{F})$*, is the smallest t such that $x^t \in \mathrm{Sep}(\mathbb{K}/\mathbb{F})$ for every $x \in \mathbb{K}$.* ◊

Remark 6.3.3. *The above definition deviates slightly from the standard definition in texts on field theory. However, this is the definition used by Pandey, Saxena and Sinhababu* [PSS18] *in their criterion and we stick with it here.* ◊

We would like to extend this definition to non-algebraic extensions.

Let $\{f_1, \ldots, f_m\}$ be a set of polynomials over $\mathbb{F}$. We will be interested in the extension $\mathbb{F}(\mathbf{x}) = \mathbb{F}(x_1, \ldots, x_n)$ over $\mathbb{F}(f_1, \ldots, f_m)$. Suppose $\{f_1, \ldots, f_k\}$ is a separable transcendence basis of $\{f_1, \ldots, f_m\}$. Using the matroid property of algebraically independent polynomials, there exists $x_{i_{k+1}}, \ldots, x_{i_n}$ such that $\{f_1, \ldots, f_k, x_{i_{k+1}}, \ldots, x_{i_n}\}$ is algebraically independent as well.

Now, since $\mathbb{F}(\mathbf{x})$ is algebraic over $\mathbb{F}(f_1, \ldots, f_k, x_{i_{k+1}}, \ldots, x_{i_n})$, we can talk about the inseparable degree of this algebraic extension. We use this to define a suitable notation of inseparable degree for a set of algebraically independent polynomials.

Definition 6.3.4 (Inseparable degree of a set of polynomials)**.** *Let $\mathbf{f} = \{f_1, \ldots, f_m\}$ be a set of polynomials over a field $\mathbb{F}$ of characteristic p. For a set $S \subseteq [n]$, define $\mathbf{x}_S = \{x_i \ : \ i \in S\}$. We shall define* $\mathsf{insep\text{-}deg}(\{f_1, \ldots, f_k\})$ *to be*

$$\min \left\{ \mathsf{insep\text{-}deg}\left(\mathbb{F}(\mathbf{x})/\mathbb{F}(\mathbf{f}, \mathbf{x}_S)\right) \ : \ \begin{array}{c} |S| = n - \mathsf{algrank}(\mathbf{f}) \text{ and} \\ \mathbb{F}(\mathbf{f}, \mathbf{x}_S)/\mathbb{F}(\mathbf{f}) \text{ is purely transcendental} \end{array} \right\}$$ ◊

Intuitively, every extension can be thought of as purely transcendental, followed by a separable algebraic, followed by a purely inseparable algebraic extension. The above definition used the inseparable degree of the purely inseparable part of this in the general case. We again note that this definition is non-standard, but is sufficient for our purposes and the criterion of Pandey, Saxena and Sinhababu [PSS18]

An important point to keep in mind is that when we are working over fields of characteristic zero, since every irreducible polynomial is separable, the inseparable degree of any set of polynomials is always 1.

With this background, we are now ready to state the criterion for algebraic independence over fields of finite characteristic. Similar to the Jacobian Criterion, Pandey, Saxena and Sinhababu [PSS18] reduce the problem of checking algebraic independence to that of checking linear independence. However, their criterion is slightly more subtle in the sense that we will have to check the linear independence of a set of vectors modulo a large subspace.

The Criterion

The key insight of Pandey *et al.* [PSS18] was to observe that the rows of the Jacobian matrix, which are first order partial derivatives, are the linear terms present in the Taylor expansion of $f(\mathbf{x})$ around a generic point $\mathbf{z}$.

In their work, they study higher order terms of the Taylor expansion around a generic point, and show that it is enough to look at terms up to the inseparable degree of the set of polynomials given. This allows them to characterise the failure of the Jacobian criterion and also come up with a modified criterion that works over all fields.

To state their criterion succinctly, they define a new operator based on the notion of *Hasse derivatives* of polynomials.

Taylor Expansion and Hasse Derivatives Let $\mathcal{H}_t(f) := \deg_{\leq t}(f(\mathbf{x}+\mathbf{z}) - f(\mathbf{z}))$, where $\deg_{\leq t}$ restricts to just those monomials in $\mathbf{x}$ of degree at most t. Note that $\mathcal{H}_t(f)$ does not have a constant term and this would become useful in the criterion.

The operator $\mathcal{H}_t(f)$ can be thought of as a vector over the field $\mathbb{F}(\mathbf{z})$ whose coordinates are indexed by monomials $\mathbf{x}^{\mathbf{e}}$ of degree at most t. The entry in the coordinate $\mathbf{x}^{\mathbf{e}}$ of $\mathcal{H}_t(\mathbf{f})$ is the corresponding *Hasse derivative* of f evaluated at $\mathbf{z}$:

$$\frac{|\mathbf{e}|!}{e_1!e_2!\cdots e_n!} \cdot \left(\frac{\partial^{|\mathbf{e}|} f}{\partial x_1^{e_1} \cdots \partial x_n^{e_n}}\right)(\mathbf{z}).$$

The operator $\mathcal{H}_t$ however, as defined above, is indexed by t. Pandey *et al.* [PSS18] show that the correct value of t to work with is the *inseparable degree* of the given set of polynomials.

Formally, we have the following statement.

Theorem 6.3.5 ([PSS18]). *Let $\{f_1, \ldots, f_k\} \in \mathbb{F}[x_1, \ldots, x_n]$ have inseparable degree t. Further, for a generic point $\mathbf{z}$, let $\mathcal{H}_t(f_i) = \deg_{\leq t}(f_i(\mathbf{x}+\mathbf{z}) - f_i(\mathbf{z}))$. Then, they are algebraically dependent if and only if there exists a* non-zero *$(\alpha_1, \ldots, \alpha_k) \in \mathbb{F}(\mathbf{z})^k$ such that*

$$\sum_{i=1}^{k} \alpha_i \cdot \mathcal{H}_t(f_i) = 0 \mod \langle \mathcal{H}_t(f_1), \ldots, \mathcal{H}_t(f_k) \rangle^{\geq 2}_{\mathbb{F}(\mathbf{z})} + \langle \mathbf{x} \rangle^{t+1}.$$

We note that at least one direction of this theorem can be slightly generalised.

A Slight Generalisation

We state this generalisation formally here, since we will need it in the next chapter. A proof is given for the sake of completeness, but we note that the steps are almost identical to those in [PSS18].

Lemma 6.3.6. *Let $\mathbb{F}$ be an algebraically closed field and $\mathbb{K}$ be an extension field of $\mathbb{F}$. Further, suppose $\{g_1, \ldots, g_k\}$ is a set of n-variate polynomials in $\mathbb{K}[\mathbf{y}]$ that are $\mathbb{F}$-algebraically dependent. Also, for a generic point $\mathbf{v}$, let*

$$\mathcal{H}_t(g_i) = \deg_{\leq t}(g_i(\mathbf{y}+\mathbf{v}) - g_i(\mathbf{v}))$$

Then for any positive integer t, there exists $(\alpha_1, \ldots, \alpha_k) \in \mathbb{F}(\mathbf{g}(\mathbf{v}))^k \setminus \{\mathbf{0}\}$ such that

$$\sum_{i=1}^{k} \alpha_i \mathcal{H}_t(g_i) \equiv 0 \mod \langle \mathcal{H}_t(g_1), \ldots, \mathcal{H}_t(g_k) \rangle^{\geq 2}_{\mathbb{F}(\mathbf{g}(\mathbf{v}))} + \langle \mathbf{y} \rangle^{t+1}$$

Proof. Suppose $\{g_1, \ldots, g_k\}$ are $\mathbb{F}$-algebraically dependent. Then by standard properties of transcendence bases [Kna07, Theorem 7.20 and 7.18], we have that there is an $\mathbb{F}$-algebraically independent subset of $\{g_1, \ldots, g_k\}$, of size $r < k$, that forms a separable transcendence basis.

Without loss of generality, let that subset be $\{g_1, \ldots, g_r\}$.

Let $A \in \mathbb{F}[u_0, u_1, \ldots, u_r]$ be the minimal annihilating polynomial for the set of polynomials $\mathbf{g} = \{g_0, g_1, \ldots, g_r\}$ where $g_0 := g_{r+1}$. Now since $A(\mathbf{g}) = 0$, for formal

variables $\mathbf{v}$, we have $A(\mathbf{g}(\mathbf{y}+\mathbf{v}))=0$. Also, from the definition of $\mathcal{H}_t(g)$, we have that

$$g_j(\mathbf{y}+\mathbf{v}) = g_j(\mathbf{v}) + \mathcal{H}_t(g_j) \mod \langle \mathbf{y} \rangle^{t+1}$$

for any $j=0,\ldots,r$. Hence,

$$A(g_0(\mathbf{v})+\mathcal{H}_t(g_0),\ldots,g_r(\mathbf{v})+\mathcal{H}_t(g_r)) = 0 \mod \langle \mathbf{y} \rangle^{t+1}.$$

Using Taylor expansion, we get

$$\begin{aligned} A(g_0(\mathbf{v})+\mathcal{H}_t(g_0),\ldots,g_r(\mathbf{v})+\mathcal{H}_t(g_r)) &= \sum_{\mathbf{e}\geq 0} (\partial_{\mathbf{u}^\mathbf{e}} A)_{\mathbf{u}=\mathbf{g}(\mathbf{v})} \cdot (\mathcal{H}_t(\mathbf{g}))^{\mathbf{e}} \\ &= A(\mathbf{g}(\mathbf{v})) + \sum_{i=0}^{r} (\partial_{u_i} A)_{\mathbf{u}=\mathbf{g}(\mathbf{v})}\, \mathcal{H}_t(g_i) \\ &\qquad \mod \langle \mathcal{H}_t(g_0),\ldots,\mathcal{H}_t(g_r)\rangle^{\geq 2}_{\mathbb{F}(\mathbf{g}(\mathbf{v}))} + \langle \mathbf{y} \rangle^{t+1} \end{aligned}$$

where the last equality crucially used the fact that the coefficients of A are from $\mathbb{F}$ and hence the linear combinations of $\langle \mathcal{H}_t(\mathbf{g}) \rangle^{\geq 2}$ are over $\mathbb{F}(\mathbf{g}(\mathbf{v}))$.

Observe that $A(\mathbf{g}(\mathbf{v}))=0$. Furthermore, since $\{g_1,\ldots,g_r\}$ forms a separable basis, we have that $\partial_{u_0} A$ is a nonzero polynomial. Hence $\partial_{u_0}(A(\mathbf{g}(\mathbf{v}))) \neq 0$, as A is the minimal degree annihilator for $\mathbf{g}$. Therefore, we have a nonzero vector $(\alpha_1,\ldots,\alpha_k) \in (\mathbb{F}(\mathbf{g}(\mathbf{v})))^k$ such that

$$\sum_{i=1}^{k} \alpha_i \mathcal{H}_t(g_i) \equiv 0 \mod \langle \mathcal{H}_t(g_1),\ldots,\mathcal{H}_t(g_k)\rangle^{\geq 2}_{\mathbb{F}(\mathbf{g}(\mathbf{v}))} + \langle \mathbf{y} \rangle^{t+1} \qquad \square$$

A Different Perspective

Let $\mathcal{U}_t(\mathbf{f}) = \mathcal{U}_t(f_1,\ldots,f_k)$ denote the subspace

$$\langle \mathcal{H}_t(\mathbf{f}) \rangle^{\geq 2}_{\mathbb{F}(\mathbf{z})} = \langle \mathcal{H}_t(f_1),\ldots,\mathcal{H}_t(f_k)\rangle^{\geq 2}_{\mathbb{F}(\mathbf{z})} \bmod \langle \mathbf{x} \rangle^{t+1}.$$

Then, for any $h \in \mathcal{U}_t(\mathbf{f})$, we define the modified Jacobian matrix as follows.

$$\mathsf{PSSJac}_t(\mathbf{f},h) = \begin{bmatrix} \mathcal{H}_t(f_1)+h \\ \mathcal{H}_t(f_2) \\ \vdots \\ \mathcal{H}_t(f_k) \end{bmatrix}.$$

The columns of this matrix are indexed by monomials in $\mathbf{x}$ and entries in the column indexed by $\mathbf{x}^{\mathbf{e}}$ are the coefficient of $\mathbf{x}^{\mathbf{e}}$ in the corresponding rows.

An alternative statement for the PSS criterion is thus, the following.

Theorem 6.3.7 (Alternate Statement for the PSS-criterion)**.** *Let $\{f_1, \ldots, f_k\}$ be a set of n-variate polynomials over a field $\mathbb{F}$ with inseparable degree t. Then, they are algebraically independent if and only if for every $h \in \mathcal{U}_t(\mathbf{f})$, $\mathsf{PSSJac}_t(\mathbf{f}, h)$ is full rank.*

We note that Lemma 6.3.6 can also be viewed from a similar perspective. Let $\mathcal{V}_t(g_1, \ldots, g_k)$ denote the subspace $\langle \mathcal{H}_t(g_1), \ldots, \mathcal{H}_t(g_k) \rangle^{\geq 2}_{\mathbb{F}(\mathbf{g}(\mathbf{v}))} \bmod \langle \mathbf{y} \rangle^{t+1}$. An alternate statement for the lemma is then the following.

Lemma 6.3.8 (Alternate Statement for Lemma 6.3.6)**.** *Let $\mathbb{F}$ any field and $\mathbb{K}$ be an extension field of $\mathbb{F}$. If $\{g_1, \ldots, g_k\}$ is a set of n-variate polynomials in $\mathbb{K}[\mathbf{y}]$ that are $\mathbb{F}$-algebraically dependent, then for any positive integer t, there exists $h' \in \mathcal{V}_t(g_1, \ldots, g_k)$ such that $\mathsf{PSSJac}_t(\mathbf{g}, h')$ is not full rank.*

We are now finally ready to talk about the complexity of testing algebraic independence. Recall that over fields of characteristic zero, the Jacobian criterion leads to a randomised polynomial time algorithm for this problem. Further, this algorithm only has one-sided error, therefore placing the problem of testing algebraic independence over fields of characteristic zero in RP. However, over fields of finite characteristic, the upper bounds known are quite weak and it remains wide open whether the problem is in RP over such fields as well [DGW09].

6.4 Testing Algebraic Independence Over Fields of Finite Characteristic

The trivial upper bound known for the problem of algebraic independence testing is PSPACE. This is due to a classical result of Perron [Per27], who showed that given a set of algebraically dependent polynomials, there exists an annihilator for these polynomials whose degree is upper bounded by the product of their degrees.

This upper bound was brought down to $\mathsf{NP}^{\#\mathsf{P}}$ due to the work of Mittman, Saxena and Scheiblechner [MSS14]. In a recent work of Guo, Saxena and Sinhababu [GSS19], the complexity was further brought down to $\mathsf{AM} \cap \mathsf{co} - \mathsf{AM}$. This makes it unlikely that the algebraic independence testing problem is NP-complete (under

standard complexity theoretic assumptions) [AB09]. However, it is unclear if we can get an improved algorithm using the ideas in [GSS19].

The PSS Criterion (Theorem 6.3.7) on the other hand, naturally suggests a randomised algorithm [PSS18, Section 4]. Further, given n polynomials as circuits of size s each, if the inseparable degree of these polynomials is t, then this algorithm runs in time $\mathrm{poly}(s, \binom{t+n}{t})$. Note that in the setting when the inseparable degree is constant, the algorithm is efficient since the running time is $O(n^t)$.

We will use this fact later, in chapter 7, to construct *faithful homomorphisms* in this setting. But before that, we look at another case where we have an efficient randomised algorithm for checking algebraic independence among polynomials over fields of finite characteristic. The case when all the polynomials are quadratics.

Our Result: Solving The Quadratic Case

Suppose we are given n quadratic polynomials over n variables. We show that in this case, the Jacobian Criterion holds in more generality. That is, the polynomials are algebraically independent if and only if the Jacobian of these polynomials has full rank, as long as the underlying field does not have characteristic two.

Theorem 6.4.1. *Let $\{f_1, \ldots, f_n\} \subseteq \mathbb{F}[x_1, \ldots, x_n]$ be a set of quadratic polynomials with $\mathrm{char}(\mathbb{F}) \neq 2$. If $f_1, \ldots, f_n$ are algebraically independent, then $\det(\mathbf{J}_{\mathbf{x}}(\mathbf{f})) \neq 0$.*

Proof. Let us assume, without loss of generality, that the given polynomials are constant free.[1] We now shift the polynomials by a random point $\mathbf{a}$ to get $g_i(\mathbf{x}) = f_i(\mathbf{x} + \mathbf{a}) - f_i(\mathbf{a})$ for every i. Then, $g_1, \ldots, g_n$ can be written as $g_i = l_i + q_i$, where q_is are quadratic forms and l_is are linear forms.

Now, suppose $\det(\mathbf{J}_{\mathbf{x}}(\mathbf{f})) = 0$. That would mean $g_1, \ldots, g_n$ are linearly dependent mod $\langle \mathbf{x} \rangle^2$. Since $g_i = l_i + q_i$ in this case, what it means is that the set $\{l_1, \ldots, l_n\}$ are linearly dependent. Without loss of generality, let the rank of $\{l_1, \ldots, l_n\}$ be $n - 1$ and let l_n be linearly dependent on the rest. Further, there must be some $i \in [n]$ such that x_i is linearly independent of $\{l_1, \ldots, l_{n-1}\}$. Without loss of generality, we assume that it is x_n.

We first make the following claim.

[1] $\{f_1, \ldots, f_n\}$ are algebraically independent if and only if $\{f_1 - f_1(\mathbf{0}), \ldots, f_n - f_n(\mathbf{0})\}$ are algebraically independent since the $\mathcal{H}$ operator does not distinguish these two cases. Further, the Jacobian also does not change and so we are in exactly the same situation.

Claim 6.4.2. *If $l_1, \ldots, l_{n-1}, x_n$ are linearly independent, then the matrix M defined as*

$$M_{ij} = \mathsf{coeff}_{x_j}(l_i)$$

for $i, j \in [n-1]$ is invertible.

Proof. Suppose the above claim is false. Then, there would be $\alpha_i, \ldots, \alpha_{n-1}$ such that $\sum_{j=1}^{n-1} \alpha_j M_j = 0$, where M_j is the jth row of M. However, since $\{l_1, \ldots, l_{n-1}\}$ are algebraically independent, $\sum_{j=1}^{n-1} \alpha_j l_j \neq 0$.

The two above statements together implies that for some α,

$$\sum_{j=1}^{n-1} \alpha_j l_j = \alpha x_n,$$

which contradicts the assumption that $l_1, \ldots, l_{n-1}, x_n$ are linearly independent. □

Now since M is invertible, for $b_i = -\mathsf{coeff}_{x_n}(l_i)$, the system $M \times \mathbf{y} = \mathbf{b}$ has a solution. Let $\{\alpha_1, \ldots, \alpha_{n-1}\}$ be the solution.

On the other hand, since $f_1, \ldots, f_n$ are algebraically independent it must be the case that $\{x_n, f_1, \ldots, f_n\}$ are algebraically dependent. Further, if p^k is the inseparable degree of the extension $\mathbb{F}(x_n, f_1, \ldots, f_n)/\mathbb{F}(f_1, \ldots, f_n)$, then it must be the case that $\{f_1, \ldots, f_n\}$ is a separable transcendence basis for the algebraically dependent set $\left\{x_n^{p^k}, f_1, \ldots, f_n\right\}$. Let $A \in \mathbb{F}[u_0, u_1, \ldots, u_n]$ be the minimal annihilating polynomial for $\left\{x_n^{p^k}, f_1, \ldots, f_n\right\}$. Then, $A((x_n + a_n)^{p^k}, f_1(\mathbf{x}+\mathbf{a}), \ldots, f_n(\mathbf{x}+\mathbf{a})) = A(x_n^{p^k} + a_n^{p^k}, f_1(\mathbf{x}+\mathbf{a}), \ldots, f_n(\mathbf{x}+\mathbf{a})) = 0$ where $\mathbf{a}$ is as defined earlier. For ease of notation, let us denote $x_n^{p^k}$ by $f_0(\mathbf{x})$ and define $g_0(\mathbf{x}) = f_0(\mathbf{x}+\mathbf{a}) - f_0(\mathbf{a}) = x_n^{p^k}$.

Now, since $f_i(\mathbf{x}+\mathbf{a}) = f_i(\mathbf{a}) + g_i(\mathbf{x})$ for every $i \in \{0, \ldots, n\}$,

$$A(f_0(\mathbf{a}) + g_0(\mathbf{x}), f_1(\mathbf{a}) + g_1(\mathbf{x}), \ldots, f_n(\mathbf{a}) + g_n(\mathbf{x})) = 0.$$

Using Taylor expansion, we get

$$\begin{aligned} A(f_0(\mathbf{a}) + g_0(\mathbf{x}), \ldots, f_n(\mathbf{a}) + g_n(\mathbf{x})) &= \sum_{\mathbf{e} \geq 0} \left(\partial_{\mathbf{u}^\mathbf{e}} A\right)_{\mathbf{u}=\mathbf{f}(\mathbf{a})} \cdot (\mathbf{g}(\mathbf{x}))^\mathbf{e} \\ &= A(\mathbf{f}(\mathbf{a})) + \sum_{i=0}^{r} \left(\partial_{u_i} A\right)_{\mathbf{u}=\mathbf{f}(\mathbf{a})} \cdot g_i(\mathbf{x}) + \sum_{\mathbf{e} \geq 2} \left(\partial_{\mathbf{u}^\mathbf{e}} A\right)_{\mathbf{u}=\mathbf{f}(\mathbf{a})} \cdot (\mathbf{g}(\mathbf{x}))^\mathbf{e}. \end{aligned}$$

Observe that $A(\mathbf{f}(\mathbf{a})) = 0$. Furthermore, since $\{f_1, \ldots, f_r\}$ forms a separable basis, we have that $\partial_{u_0} A$ is a nonzero polynomial. Hence $\partial_{u_0}(A(\mathbf{f}(\mathbf{a}))) \neq 0$, as A is the minimal degree annihilator for $\mathbf{g}$.

Thus we have

$$\begin{aligned} 0 &= \partial_{u_0}(A(\mathbf{f}(\mathbf{a}))) \cdot x_n^{p^k} + A(\mathbf{f}(\mathbf{a})) + \sum_{i=1}^{r} (\partial_{u_i} A)_{\mathbf{u}=\mathbf{f}(\mathbf{a})} \cdot g_i(\mathbf{x}) \\ &\qquad + \sum_{\mathbf{e} \geq 2} (\partial_{\mathbf{u}^\mathbf{e}} A)_{\mathbf{u}=\mathbf{f}(\mathbf{a})} \cdot (\mathbf{g}(\mathbf{x}))^\mathbf{e} \\ &= \partial_{u_0}(A(\mathbf{f}(\mathbf{a}))) \cdot x_n^{p^k} + A(\mathbf{f}(\mathbf{a})) + \sum_{i=1}^{r} (\partial_{u_i} A)_{\mathbf{u}=\mathbf{f}(\mathbf{a})} \cdot g_i(\mathbf{x}) \\ &\qquad + \sum_{\mathbf{e} \geq 2} (\partial_{\mathbf{u}^\mathbf{e}} A)_{\mathbf{u}=\mathbf{f}(\mathbf{a})} \cdot (g_1^{e_1} \cdots g_n^{e_n}) \mod \langle x \rangle^{p^k+1}. \end{aligned}$$

for $\partial_{u_0}(A(\mathbf{f}(\mathbf{a}))) \neq 0$. In other words,

$$x_n^{p^k} = \mathcal{R}_\mathbf{a}(g_1, \ldots, g_n) \mod \langle \mathbf{x} \rangle^{\geq p^k+1}$$

for some $\mathcal{R}_\mathbf{a} \in \mathbb{F}[y_1, \ldots y_n]$ that depends on the random point $\mathbf{a}$. That is,

$$x_n^{p^k} = \mathcal{R}_\mathbf{a}(g_1, \ldots, g_n) + \mathcal{E}(\mathbf{x})$$

where the degree of every monomial in $\mathcal{E}(\mathbf{x})$ is at least $p^k + 1$.

Since the left hand side of the equation depends only on x_n, we should be able to substitute the other variables $x_1, \ldots, x_{n-1}$ to any value and the equation should still be satisfied. Let us thus make the following substitution.

$$x_i = \alpha_i x_n \qquad \text{for every } i \in [n-1]$$

Since the substitution preserves the degree of each monomial, by our choice of $\{\alpha_1, \ldots, \alpha_{n-1}\}$, the equation now becomes

$$x_n^{p^k} = \mathcal{R}_\mathbf{a}(q_1, \ldots, q_n) + \mathcal{E}(x_n).$$

Here every monomial in $\mathcal{E}$ has degree at least $p^k + 1$.

Now note that the right hand side of the equation can never generate a monomial that has degree at most p^k and is odd. Hence, for $p \neq 2$, the above equation cannot be true. This shows that our initial assumption that $\det(\mathbf{J}_\mathbf{x}(f_1, \ldots, f_n)) = 0$ must be false. □

For higher degrees however, no algorithm better than the one suggested by the PSS criterion is known. When the inseparable degree is non-constant, this algorithm fails to be efficient. Therefore, any progress made towards giving an efficient randomised algorithm for checking algebraic independence over fields of finite characteristics when the inseparable degree is non-constant, would be extremely interesting.

7 Constructing Faithful Homomorphisms

Algebraic independence shares a lot of similarities with linear independence due to the matroid structure. One natural task is to find a *rank-preserving transformation* in this setting. This is defined by what are called *faithful homomorphisms*.

Intuitively, suppose we are given a set of n-variate polynomials $\{f_1, \ldots, f_m\}$ that have algebraic rank k. Then a map φ from this set to the set of k-variate polynomials is said to be faithful if the algebraic rank of $\{\varphi(f_1), \ldots, \varphi(f_m)\}$ is also k.

Apart from being a natural task, constructing faithful homomorphisms also has applications in polynomial identity testing, one of the central algorithmic questions in algebraic circuit complexity. Given an algebraic circuit $\mathcal{C}$, polynomial identity testing is the task of checking whether $\mathcal{C}$ computes the identically zero polynomial.

7.1 Faithful Homomorphisms and Polynomial Identity Tests

Let us now look at the formal definition of faithful homomorphisms.

Definition 7.1.1 (Faithful homomorphisms [BMS13])**.** *Let* $\mathbf{f} = \{f_1, \ldots, f_m\} \subseteq \mathbb{F}[\mathbf{x}]$ *be a set of polynomials. If* $\mathbb{K}$ *is an extension field of* $\mathbb{F}$*, a homomorphism* $\Phi : \mathbb{F}[\mathbf{x}] \to \mathbb{K}[\mathbf{y}]$ *is said to be an* $\mathbb{F}$-faithful *homomorphism for* $\{f_1, \ldots, f_m\}$ *if*

$$\mathsf{algrank}_{\mathbb{F}}\{f_1, \ldots, f_m\} = \mathsf{algrank}_{\mathbb{F}}\{\Phi(f_1), \ldots, \Phi(f_m)\}. \qquad \Diamond$$

Ideally, we would like a faithful homomorphism with $|\mathbf{y}| \approx \mathsf{algrank}\{\mathbf{f}\}$ and $\mathbb{K} = \mathbb{F}$. Beecken, Mittmann and Saxena [BMS13] showed that a *generic* $\mathbb{F}$-linear homomorphism to $\mathsf{algrank}(\mathbf{f})$ many variables would be $\mathbb{F}$-faithful with high probability.

One important consequence of faithful homomorphisms is that they preserve nonzeroness of any polynomial composition of $f_1, \ldots, f_m$.

Lemma 7.1.2 ([BMS13; Agr+16]). *Suppose $f_1, \ldots, f_m \in \mathbb{F}[x_1, \ldots, x_n]$ and Φ is an $\mathbb{F}$-faithful homomorphism for $\{f_1, \ldots, f_m\}$. Then, for any circuit $C(z_1, \ldots, z_m) \in \mathbb{F}[z_1, \ldots, z_m]$, we have*

$$C(f_1, \ldots, f_m) = 0 \Leftrightarrow C(\Phi(f_1), \ldots, \Phi(f_m)) = 0.$$

Thus, constructing explicit faithful homomorphisms can also be used for polynomial identity testing or PIT, where the goal is to design a deterministic algorithm that runs in time polynomial in the size of the circuit.

There are two types of PIT algorithms, *whitebox* and *blackbox* — in the blackbox setting, we are only provided evaluation access to the circuit and some of its parameters (such as degree, number of variables, size etc.). Thus blackbox PIT algorithms for a class $\mathcal{C}$ is equivalent to constructing a *hitting set*, which is a small list of points in $S \subset \mathbb{F}^n$ such that any nonzero polynomial $f \in \mathcal{C}$ is guaranteed to evaluate to a nonzero value on some $\mathbf{a} \in S$.

It follows from Lemma 7.1.2 that if we can construct explicit $\mathbb{F}$-faithful homomorphisms for a set $\{f_1, \ldots, f_m\}$ whose algebraic rank is $k \ll n$, then we have a *variable reduction* that preserves the nonzeroness of any composition $C(f_1, \ldots, f_m)$.

This approach was used by Beecken, Mittmann and Saxena [BMS13] and Agrawal, Saha, Saptharishi, Saxena [Agr+16], in the characteristic zero setting, to design identity tests for several subclasses by constructing faithful homomorphisms for $\{f_1, \ldots, f_m\}$ with algebraic rank at most $k = O(1)$, when

- each f_i is a sparse polynomial,
- each f_i is a product of multilinear, variable disjoint, sparse polynomials,
- each f_i is a product of linear polynomials,

and further generalisations.

All the above constructions crucially depend on the fact that the rank of the Jacobian captures algebraic independence. However, this fact is true only over fields of characteristic zero and hence all the above results no longer hold over fields of finite characteristic. Following up on the criterion of Pandey, Saxena and Sinhababu [PSS18] for algebraic independence over such fields, we extend the results of Beecken *et al.* [BMS13] and Agrawal *et al.* [Agr+16] to construct faithful homomorphisms over fields for some restricted settings. This then allows us to give efficient polynomial identity tests for certain circuit classes.

7.2 Our Results

Let us first state our results formally.

Theorem 7.2.1. *Let $f_1, \ldots, f_m \in \mathbb{F}[x_1, \ldots, x_n]$ be such that* algrank $\{f_1, \ldots, f_m\} = k$ *and the inseparable degree is t. If t and k are bounded by a constant, then we can construct a polynomial (in the input length) sized list of homomorphisms of the form $\Phi : \mathbb{F}[\mathbf{x}] \to \mathbb{F}(s)[y_0, y_1, \ldots, y_k]$ such that at least one of them is guaranteed to be to $\mathbb{F}$-faithful for the set $\{f_1, \ldots, f_m\}$, in the following two settings:*

- *each of the f_i's are sparse polynomials,*
- *each of the f_i's are products of variable disjoint, multilinear, sparse polynomials.*

Prior to this, construction of faithful homomorphisms over finite fields was known only in the setting when each f_i has small individual degree [BMS13]. Over characteristic zero fields, the inseparable degree is always 1 and hence the faithful homomorphisms constructed in [BMS13], [Agr+16] over such fields can be viewed as special cases of our constructions.

Our theorem also holds for a few other models studied by Agrawal *et al.*[Agr+16] (for instance, occur-k products of sparse polynomials).

We mention the above two models just as an illustration of lifting the recipe for faithful homomorphisms from [BMS13; Agr+16] to the finite characteristic setting.

As corollaries, we get efficient PIT algorithms for these models.

Corollary 7.2.2. *If $\{f_1, \ldots, f_m\} \in \mathbb{F}[x_1, \ldots, x_n]$ is a set of s-sparse polynomials with algebraic rank k and inseparable degree t where $k, t = O(1)$. Then, for the class of polynomials of the form $C(f_1, \ldots, f_m)$ for any polynomial $C(z_1, \ldots, z_m) \in \mathbb{F}[\mathbf{z}]$, there is an explicit hitting set of size $(s \cdot \deg(C))^{O(1)}$.*

Corollary 7.2.3. *Let $\mathcal{C} = \sum_{i=1}^{m} T_i$ be a depth-4 multilinear circuit of size s, where each T_i is a product of variable-disjoint, s-sparse polynomials. Suppose $\{T_1, \ldots, T_m\} \in \mathbb{F}[x_1, \ldots, x_n]$ is a set of polynomials with algebraic rank k and inseparable degree t where $k, t = O(1)$. Then, for the class of polynomials of the form $C(T_1, \ldots, T_m)$ for any polynomial $C(z_1, \ldots, z_m) \in \mathbb{F}[\mathbf{z}]$, there is an explicit hitting set of size $(s \cdot \deg(C))^{O(1)}$.*

We now compare these PIT results with those in related works.

Comparison with the PIT of Pandey *et al.* [PSS18] Pandey *et al.* [PSS18] also gives a PIT result for circuits of the form $\sum_i (f_{i,1} \cdots f_{i,m})$ where $\mathsf{algrank}\{f_{i,1}, \ldots, f_{i,m}\} \leq k$ for every i and each $f_{i,j}$ is a degree d polynomial in $\mathbb{F}[x_1, \ldots, x_n]$. They extend the result of Kumar and Saraf [KS17a] to arbitrary fields by giving quasi-polynomial time hitting sets if kd is at most poly-logarithmically large. Corollary 7.2.3 however is incomparable to their result for the following reasons:

- The algebraic rank bound in the case of [PSS18; KS17a] is a gate-wise bound rather than a global bound. Thus, in principle, it could be the case that $\mathsf{algrank}\{f_{i,1}, \ldots, f_{i,m}\}$ is bounded by k for each i but this would not necessarily translate to a bound on $\mathsf{algrank}\left\{\prod_j f_{i,j} \ : \ i\right\}$ as demanded in Corollary 7.2.3. Hence, in this regard, the PIT of [PSS18; KS17a] is stronger.
- In the regime when we have $\mathsf{algrank}\left\{\prod_j f_{i,j} \ : \ i\right\}$ and the inseparable degree of this set to be bounded by a constant, Corollary 7.2.3 presents an explicit hitting set of polynomial size, whereas it is unclear if [PSS18; KS17a] provide any non-trivial upper bound as this does not translate to any bound on $\mathsf{algrank}\{f_{i,1}, \ldots, f_{i,m}\}$.

On other models studied by Agrawal *et al.* [Agr+16] Our results, in its current form, do not extend directly some of the other models studied by Agrawal *et al.* [Agr+16], most notably larger depth multilinear formulas. The primary hurdle appears to be the *recursive* use of explicit faithful homomorphisms for larger depth formulas. In the characteristic p setting, unfortunately, it is unclear if a bound on the inseparable degree of the original gates can be used to obtain a bound on the inseparable degree of other sets of polynomials considered in the recursive construction of [Agr+16].

Proof Overview

We now go over the proof idea of Theorem 7.2.1. The general structure of the proof follows the outline of Agrawal *et al.* [Agr+16]'s construction of faithful homomorphisms in the characteristic zero setting. Roughly speaking, this can be described in the following steps:

Step 1 : For a *generic linear map* $\Phi : \mathbf{x} \to \mathbb{F}(s)[y_1, \ldots, y_k]$, write the Jacobian of the set of polynomials $\{f_1 \circ \Phi, \cdots, f_k \circ \Phi\}$. Thus can be described succinctly as a matrix product of the form

$$J_{\mathbf{y}}(f \circ \Phi) = \Phi(J_{\mathbf{x}}(\mathbf{f})) \cdot J_{\mathbf{y}}(\Phi(\mathbf{x})).$$

Step 2 : We know that $J_{\mathbf{x}}(\mathbf{f})$ is full rank. Ensure that $\Phi(J_{\mathbf{x}}(\mathbf{f}))$ (where Φ is applied to every entry of the matrix $J_{\mathbf{x}}(\mathbf{f})$) remains full rank. This can be done if $\mathbf{f}$'s are some structured polynomials such as sparse polynomials, or variable-disjoint products of sparse polynomials etc.

Step 3 : Choose the map Φ so as to ensure that

$$\operatorname{rank}(\Phi(J_{\mathbf{x}}(\mathbf{f})) \cdot J_{\mathbf{y}}(\Phi(\mathbf{x}))) = \operatorname{rank}(\Phi(J_{\mathbf{x}}(\mathbf{f}))).$$

This is typically achieved by choosing Φ so as to make $J_{\mathbf{y}}(\Phi(\mathbf{x}))$ a *rank-extractor*. It was shown by Gabizon and Raz [GR08] that a parametrized Vandermonde matrix has this property and this allows us to work with a homomorphism of the form (loosely speaking)

$$\Phi : x_i \mapsto \sum_{j=1}^{k} s^{ij} y_j.$$

We would like to execute essentially the same sketch over fields of finite characteristic but we encounter some immediate difficulties. The criterion of Pandey *et al.* [PSS18] over finite characteristic is more involved but it is reasonably straightforward to execute Steps 1 and 2 in the above sketch using the chain rule of (Hasse) derivatives. The primary issue is in executing Step 3 and this is for two very different reasons.

The first is that, unlike in the characteristic zero setting, the analogue of the matrix $J_{\mathbf{y}}(\Phi(\mathbf{x}))$ has many correlated entries. In the characteristic zero setting, we have complete freedom to choose Φ so that $J_{\mathbf{y}}(\Phi(\mathbf{x}))$ can be any matrix that we want. Roughly speaking, we only have $n \cdot k$ parameters to define Φ but the analogue of $J_{\mathbf{y}}(\Phi(\mathbf{x}))$ is much larger in the finite characteristic setting. Fortunately, there is just about enough structure in the matrix that we can show that it continues to have some rank-preserving properties. This is done in section 7.4.

The second hurdle comes from the subspace that we need to work with in the modified criterion. The *rank-extractor* is essentially parametrized by the variable s. In order to show that it preserves the rank of $\Phi(J_{\mathbf{x}}(\mathbf{f}))$ under right multiplication, we would like ensure that the variable s effectively does not appear in this matrix.

In the characteristic zero setting, this is done by suitable restriction on other variables to remove any dependencies on s in $\Phi(J_{\mathbf{x}}(\mathbf{f}))$. Unfortunately, in the criterion of Pandey *et al.* [PSS18], we have to work modulo some suitable subspace and these elements introduce other dependencies on s that appear to be hard to remove. Due

to this hurdle, we are unable to construct $\mathbb{F}(s)$-faithful homomorphisms even in restricted settings.

However, we observe that for the PIT applications, we are merely required to ensure that $\{f_1 \circ \Phi, \ldots, f_k \circ \Phi\}$ remain $\mathbb{F}$-algebraically independent instead of $\mathbb{F}(s)$-algebraically independent. With this weaker requirement, we can obtain a little more structure in the subspace involved and that lets us effectively execute Step 3.

We now describe some preliminary concepts in the next section, that are necessary to understand the proof. Following that, in section 7.4, we show that certain Vandermonde-like matrices have *rank-preserving properties*. We use these matrices to give a recipe of constructing faithful homomorphisms, in section 7.5, and execute this for the settings of Theorem 7.2.1 in section 7.6.

7.3 Preliminaries

We begin by defining some notations.

Notations

- We will use bold face letters such as $\mathbf{x}$ to denote a set of indexed variables $\{x_1, \ldots, x_n\}$. In most cases the size of this set would be clear from context. Extending this notation, we will use $\mathbf{x}^{\mathbf{e}}$ to denote the monomial $x_1^{e_1} \cdots x_n^{e_n}$.
- For a set of polynomials $f_1, \ldots, f_m$, we will denote by $\langle f_1, \ldots, f_m \rangle_{\mathbb{K}}$ the set of all $\mathbb{K}$-linear combinations of $f_1, \ldots, f_m$. Extending this notation, we will use $\langle f_1, \ldots, f_m \rangle_{\mathbb{K}}^{r}$ to denote the set of all $\mathbb{K}$-linear combinations of r-products $f_{i_1} \cdots f_{i_r}$ (with $i_1, \ldots, i_r \in [m]$) and $\langle f_1, \ldots, f_m \rangle_{\mathbb{K}}^{\geq r}$ similarly. In instances when we just use $\langle f_1, \ldots, f_m \rangle$, we will denote the *ideal* generated by $f_1, \ldots, f_m$.

A well known concept that we will need is that of *hitting set generators* or HSGs.

Definition 7.3.1 (Hitting Set Generators (HSG))**.** *Let $\mathcal{C}$ be a class of n-variate polynomials. A tuple of polynomials $\mathcal{G} = (G_1(\alpha), \ldots, G_n(\alpha))$ is a* hitting set generator *for $\mathcal{C}$ if for every nonzero polynomial $P(\mathbf{x}) \in \mathcal{C}$ we have $P(G_1(\alpha), \ldots, G_n(\alpha))$ is a nonzero polynomial in α.*

The degree of this generator is defined to be $\max \deg(G_i)$. $\Diamond$

Intuitively, such a tuple can be used to *generate* a hitting set for $\mathcal{C}$ by running over several instantiations of α. Also, it is well known that any hitting set can be transformed into an HSG via interpolation.

Isolating Weight Assignments

Suppose $\mathsf{wt} : \{x_i\} \to \mathbb{N}$ is a weight assignment for the variables $\{x_1, \ldots, x_n\}$. We can extend it to define the weight of a monomial as follows.

$$\mathsf{wt}(\mathbf{x}^{\mathbf{e}}) = \sum_{i=1}^{n} e_i \cdot \mathsf{wt}(x_i)$$

Definition 7.3.2. *A weight assignment* $\mathsf{wt} : \{x_i\} \to \mathbb{N}$ *is said to be isolating for a set* S *of monomials if every pair of distinct monomials in* S *receives distinct weights.* ◊

Note that if the highest degree of a monomial in S is d, then assigning the weight $\mathsf{wt}(x_i) = (d+1)^i$ is trivially isolating for S. However, in this case the weight of a monomial can become exponentially large in n.

In the case when $|S| = \mathrm{poly}(n)$, results by Klivans and Spielman [KS01] or Agrawal and Biswas [AB03] show that if $\mathsf{wt}(x_i) = (d+1)^i \mod p$, then it suffices to go over $\mathrm{poly}(n)$ many 'p's to ensure that one of these assignments isolates the monomials in S. Thus the weight of a monomial in this case is bounded by $\mathrm{poly}(n)$.

7.4 Constructing Rank Condensers Using Isolating Weight Assignments

In this section, we focus on *rank-preserving* properties of certain types of matrices. These are slight generalisations of similar properties of Vandermonde matrices that were proved by Gabizon and Raz [GR08] that would be necessary for the application to constructing faithful homomorphisms.

Lemma 7.4.1. *For a formal variable* s*, suppose we have a matrix of the form*

$$V = \begin{bmatrix} s^{w_1} & s^{2w_1} & \ldots & s^{nw_1} \\ s^{w_2} & s^{2w_2} & \ldots & s^{nw_2} \\ & & \vdots & \\ s^{w_n} & s^{2w_n} & \ldots & s^{nw_n} \end{bmatrix}$$

where $w_i < w_j$ *whenever* $i < j$*.*

If V' *is a matrix obtained from* V *by replacing some of the non-diagonal entries by zero, then* $\det(V') \neq 0$ *and furthermore* $\deg(\det(V')) = \sum_{i=1}^{n} i \cdot w_i$*.*

Proof. Since

$$\det(V') = \sum_{\sigma \in S_n} \operatorname{sgn}(\sigma) \left(\prod_{i \in [n]} V'[i, \sigma(i)] \right),$$

the monomial corresponding to σ being the identity permutation contributes a nonzero monomial of degree $\sum i \cdot w_i$. We will show that all other terms of $\det(V')$ have smaller degree.

Suppose σ is not the identity permutation, we must have $i \neq \sigma(i)$ for some index i; let i_0 be the first such index. Define j such that $\sigma(j) = i_0$ and $\pi = \sigma \circ (i_0\ j)$. Note that $\pi(i_0) = \sigma(j) = i_0$ and fixes the first i_0 indices. Furthermore, $\pi(i) = \sigma(i)$ for all $i \neq i_0, j$. Thus,

$$\begin{aligned}
\sum_{i=1}^{n} (\pi(i) - \sigma(i)) \cdot w_i &= (\pi(i_0) - \sigma(i_0)) \cdot w_{i_0} + (\pi(j) - \sigma(j)) \cdot w_j \\
&= (\sigma(j) - \sigma(i_0)) \cdot w_{i_0} + (\sigma(i_0) - \sigma(j)) \cdot w_j \\
&= (\sigma(i_0) - \sigma(j)) \cdot (w_j - w_{i_0}) > 0
\end{aligned}$$

Repeating this exercise until we reach the identity permutation, we have that the monomial contributed by the diagonal has the largest degree. □

Lemma 7.4.2. *Let A be a matrix over a field $\mathbb{F}$ with k rows and columns indexed by monomials in $\mathbf{x}$ of degree at most D that is full-rank. Further, let $w = (w_1, \ldots, w_n)$ be an isolating weight assignment for the set of degree D monomials, and let $\mathsf{wt}(\mathbf{x}^{\mathbf{e}}) = \sum_{i=1}^{n} w_i e_i$.*

Suppose M_Φ is a matrix whose rows are indexed by monomials in $\mathbf{x}$ of degree at most D, and columns indexed by pure monomials $\left\{ y_i^d \ : \ i \in \{1, \ldots, k\} \ , \ d \leq D \right\}$ *given by*

$$M_\Phi(\mathbf{x}^{\mathbf{e}}, y_i^d) = \begin{cases} s^{i \cdot \mathsf{wt}(\mathbf{x}^{\mathbf{e}})} & \text{if } \deg(\mathbf{x}^{\mathbf{e}}) = d \\ 0 & \text{otherwise} \end{cases}.$$

where s is a formal variable. Then, $\operatorname{rank}_{\mathbb{F}(s)}(A \cdot M_\Phi) = \operatorname{rank}_{\mathbb{F}}(A)$.

Proof. By the Cauchy-Binet formula, if we restrict M_Φ to a set T of k-columns, then

$$\det(A \cdot M_\Phi[T]) = \sum_{\substack{S \subseteq \text{Columns}(A) \\ |S| = k}} \det(A[S]) \cdot \det(M_\Phi[S, T])$$

Here $M_\Phi[T]$ denotes the matrix obtained by restricting M_Φ to the columns in T and $M_\Phi[S,T]$ denotes the matrix obtained by restricting $M_\Phi[T]$ to the rows in S. Similarly, $A[S]$ denotes the matrix obtained by restricting A to the columns in S.

We wish to show that the above sum is nonzero for some choice of columns T. We do that by first defining a weight function on minors of A, then proving that there is a unique nonzero minor of A of largest weight, and then choosing a set of columns T such that the degree of $\det(M_\Phi[S,T])$ coincides with this chosen weight function.

Define the *weight* of a minor of A as follows:

> Suppose the columns of the minor is indexed by $S = \{\mathbf{x}^{\mathbf{e_1}}, \ldots, \mathbf{x}^{\mathbf{e_k}}\}$ with the property that $\mathsf{wt}(\mathbf{x}^{\mathbf{e_1}}) < \mathsf{wt}(\mathbf{x}^{\mathbf{e_2}}) < \cdots < \mathsf{wt}(\mathbf{x}^{\mathbf{e_k}})$. Define the weight of this minor as
>
> $$\mathsf{wt}(S) = \sum_{i=1}^{k} i \cdot \mathsf{wt}(\mathbf{x}^{\mathbf{e_i}})$$
>
> where, recall, $\mathsf{wt}(\mathbf{x}^{\mathbf{e_i}}) = \sum_j w_j \cdot \mathbf{e_i}(j)$.

Claim 7.4.3. *There is a unique nonzero $k \times k$ minor of A of maximum weight.*

Proof. Suppose S_1 and S_2 are two different minors of A with the same weight. We will just identify S_1 and S_2 by the set of column indices for simplicity. Say S_1 has columns indexed by $\mathbf{x}^{\mathbf{e_1}}, \ldots, \mathbf{x}^{\mathbf{e_k}}$ with $\mathsf{wt}(\mathbf{x}^{\mathbf{e_1}}) < \mathsf{wt}(\mathbf{x}^{\mathbf{e_2}}) < \cdots < \mathsf{wt}(\mathbf{x}^{\mathbf{e_k}})$ and S_2 has columns indexed by $\mathbf{x}^{\mathbf{e'_1}}, \ldots, \mathbf{x}^{\mathbf{e'_k}}$ with $\mathsf{wt}(\mathbf{x}^{\mathbf{e'_1}}) < \mathsf{wt}(\mathbf{x}^{\mathbf{e'_2}}) < \cdots < \mathsf{wt}(\mathbf{x}^{\mathbf{e'_k}})$.

Suppose S_1 and S_2 agree on the first i columns, that is $\mathbf{e}_j = \mathbf{e}'_j$ for all $j \leq i$, and say $\mathsf{wt}(\mathbf{e}_{i+1}) < \mathsf{wt}(\mathbf{e}'_{i+1})$. By the matroid property, there must be some column $\mathbf{x}^{\mathbf{e}'_j}$ from S_2 that we can add to $S_1 \setminus \{\mathbf{x}^{\mathbf{e}_{i+1}}\}$ so that $S = S_1 \setminus \{\mathbf{x}^{\mathbf{e}_{i+1}}\} \cup \left\{\mathbf{x}^{\mathbf{e}'_j}\right\}$ is also a nonzero minor of A. Suppose that

$$\mathsf{wt}(\mathbf{x}^{\mathbf{e}_1}) < \cdots < \mathsf{wt}(\mathbf{x}^{\mathbf{e}_{i+r}}) < \mathsf{wt}(\mathbf{x}^{\mathbf{e}'_j}) < \mathsf{wt}(\mathbf{x}^{\mathbf{e}_{i+r+1}}) < \cdots < \mathsf{wt}(\mathbf{x}^{\mathbf{e}_k}).$$

Then the weight of S is equal to

$$\begin{aligned}
&\sum_{a=1}^{i} a \cdot \mathsf{wt}(\mathbf{x}^{\mathbf{e}_a}) + \sum_{a=i+2}^{i+r} (a-1)\,\mathsf{wt}(\mathbf{x}^{\mathbf{e}_a}) + (i+r)\,\mathsf{wt}(\mathbf{x}^{\mathbf{e}'_j}) + \sum_{a=i+r+1}^{k} a \cdot \mathsf{wt}(\mathbf{x}^{\mathbf{e}_a}) \\
&\quad > \sum_{a=1}^{i} a \cdot \mathsf{wt}(\mathbf{x}^{\mathbf{e}_a}) + (i+1) \cdot \mathsf{wt}(\mathbf{x}^{\mathbf{e}'_j}) + \sum_{a=i+2}^{k} a \cdot \mathsf{wt}(\mathbf{x}^{\mathbf{e}_a}) > \sum_{a=1}^{k} a \cdot \mathsf{wt}(\mathbf{x}^{\mathbf{e}_a}).
\end{aligned}$$

That is, $\mathsf{wt}(S) > \mathsf{wt}(S_1)$.

Hence, there cannot be two different nonzero minors of A of the same weight. Thus, the nonzero minor of largest weight is unique. □

We will now choose k columns from M_Φ as follows in such a way that the degree of the corresponding determinant agrees with the weight function.

Note that the matrix M_Φ has a natural block-diagonal structure based on the degree of the monomials indexing the rows and columns.

> Let S_0 be the unique $k \times k$ minor of A having maximum weight. Further, assume its columns are indexed by $\mathbf{x}^{\mathbf{e_1}}, \ldots, \mathbf{x}^{\mathbf{e_k}}$ with $\mathsf{wt}(\mathbf{x}^{\mathbf{e_1}}) < \mathsf{wt}(\mathbf{x}^{\mathbf{e_2}}) < \ldots < \mathsf{wt}(\mathbf{x}^{\mathbf{e_k}})$. Let $d_i = \deg(\mathbf{x}^{\mathbf{e}_i}) = \sum_j (\mathbf{e}_i)_j$.
>
> Then choose the columns $T = \left\{ y_1^{d_1}, y_2^{d_2}, \ldots, y_k^{d_k} \right\}$ of the matrix M'_Φ.

By Lemma 7.4.1, for any set of $S' \subseteq \mathrm{Columns}(A)$, we have $\deg(\det(M_\Phi[S', T])) \leq \mathsf{wt}(S')$ and furthermore we also have $\deg(M_\Phi[S_0, T]) = \mathsf{wt}(S_0)$ as we chose the columns T to ensure that the main diagonal of the sub-matrix has only nonzero elements. Hence,

$$\det(A \cdot M_\Phi[T]) = \sum_{\substack{S \subseteq \mathrm{Columns}(A) \\ |S|=k}} \det(A[S]) \cdot \det(M_\Phi[S, T]) \neq 0$$

since the contribution from $A[S_0] \det(M_\Phi[S_0, T])$ is the unique term of highest degree and so cannot be cancelled. □

7.5 Constructing Explicit Faithful Homomorphisms

We will be interested in applying a map $\Phi : \mathbb{F}[\mathbf{x}] \to \mathbb{F}(s)[\mathbf{y}]$ and study the transformation of the PSS-Jacobian. Since the entries of the PSS-Jacobian involve $\mathcal{H}_t(f(\mathbf{x})) = \deg_{\leq t}(f(\mathbf{x}+\mathbf{z}) - f(\mathbf{z}))$, we would need to also work with $\mathcal{H}_t(g(\mathbf{y}))$ where $g(\mathbf{y}) = f \circ \Phi$. To make it easier to follow, we shall use a different name for the variables in the two cases. So let

$$\mathcal{H}_t(f(\mathbf{x})) := \deg_{\leq t}(f(\mathbf{x}+\mathbf{z}) - f(\mathbf{z})), \quad \mathcal{H}_t(g(\mathbf{y})) := \deg_{\leq t}(g(\mathbf{y}+\mathbf{v}) - g(\mathbf{v})).$$

Recipe for Constructing Faithful Homomorphisms

We are finally ready to construct faithful homomorphisms.

Let $f_1, \ldots, f_m \in \mathbb{F}[x_1, \ldots, x_n]$ be polynomials with algebraic rank k and inseparable degree t. We will work with linear transformations of the following form.

$$\Phi : x_i \mapsto a_i y_0 + \sum_{j=1}^{k} s^{w_i \cdot j} y_j, \quad \text{for all } i \in [n],$$
$$\Phi_z : z_i \mapsto a_i v_0 + \sum_{j=1}^{k} s^{w_i \cdot j} v_j, \quad \text{for all } i \in [n],$$

where all the variables on the RHS are formal variables. Further, define $\{g_1, \ldots, g_m\} \in \mathbb{F}[\mathbf{z}]$ as $g_i = f_i \circ \Phi$ and $\mathcal{H}_t(g_i) = \deg_{\leq t}(g_i(\mathbf{y} + \mathbf{v}) - g_i(\mathbf{v}))$. The main lemma of this section is the following *recipe*.

Lemma 7.5.1 (Recipe for faithful homomorphisms)**.** *Let $f_1, \ldots, f_m \in \mathbb{F}[\mathbf{x}]$ be polynomials such that their algebraic rank is at most k and suppose the inseparable degree is bounded by a constant t. Further,*

- *suppose $\mathcal{G} = (G_1(\alpha), \ldots, G_n(\alpha)) = (a_1, \ldots, a_n)$ is such that for some $\mathbf{a} \in \mathcal{G}$, the rank of $\mathsf{PSSJac}_t(\mathbf{f}, h)$ is preserved after the substitution $\mathbf{z} \to \mathbf{a}$.*
- *suppose $w : [n] \to \mathbb{N}$ is an isolating weight assignment for the set of n-variate monomials of degree at most t.*

Then, the homomorphism $\Phi : \mathbb{F}[x_1, \ldots, x_n] \to \mathbb{F}(s, \alpha)[y_0, \ldots, y_k]$ defined as

$$\Phi : x_i \mapsto y_0 G_i(\alpha) + \sum_{j=1}^{k} y_j \cdot s^{w(i)j},$$

is an $\mathbb{F}$-faithful homomorphism for the set $\{f_1, \ldots, f_m\}$.

As mentioned earlier, the rough proof sketch would be to first write the PSS-Jacobian of the transformed polynomials $\mathbf{g}$ in terms of $\mathbf{f}$, express that as a suitable matrix product, and use some *rank extractor* properties of the associated matrix, as described in section 7.4. The rest of this section will execute this sketch.

Lemma 7.5.2 (Evolution of polynomials under Φ)**.** *Let $\Phi : \mathbf{x} \to \mathbb{F}(s)[\mathbf{y}]$ and $\Phi_z : \mathbf{z} \to \mathbb{F}(s)[\mathbf{v}]$ be given as above. Further, for any polynomial $h'(a_1, \ldots, a_m) \in \mathbb{F}(\mathbf{g}(\mathbf{v}))[\mathbf{a}]$, define $h(a_1, \ldots, a_m) \in \mathbb{F}(\mathbf{f}(\mathbf{z}))[\mathbf{a}]$ as follows.*

$\mathsf{coeff}_{\mathbf{a}^\mathbf{e}}(h)$ *is obtained by replacing every occurrence of* $g_i(\mathbf{v})$ *by* $f_i(\mathbf{z})$ *in* $\mathsf{coeff}_{\mathbf{a}^\mathbf{e}}(h')$

Then, $h'(\mathcal{H}_t(g_1), \ldots, \mathcal{H}_t(g_m)) = \Phi \circ \Phi_z(h(\mathcal{H}_t(f_1), \ldots, \mathcal{H}_t(f_m)))$.

It is worth noting that the polynomial $h(a_1, \ldots, a_m)$ is independent of s. This would be crucial later on in the proof.

Proof. Firstly, note that h is well defined. This is because by the definition of $\{g_1, \ldots, g_m\}$, if $\mathsf{coeff}_{\mathbf{a}^\mathbf{e}}(h') \in \mathbb{F}(\mathbf{g}(\mathbf{v}))$ has a nonzero denominator then by replacing the $g_i(\mathbf{v})$s with $f_i(\mathbf{z})$ in it, it will continue to remain nonzero. The claim now follows essentially from the fact that Φ is linear and homogeneous in $\mathbf{y}$.

$$\begin{aligned}
\mathcal{H}_t(f \circ \Phi)(\mathbf{y}, \mathbf{v}) &= \deg_{\leq t}\left[(f \circ \Phi)(\mathbf{y} + \mathbf{v}) - (f \circ \Phi)(\mathbf{v})\right] \\
&= \deg_{\leq t}\left[f(\Phi(\mathbf{x}) + \Phi_z(\mathbf{z})) - f(\Phi_z(\mathbf{z}))\right] && \text{(by linearity in } \mathbf{y}\text{)} \\
&= \Phi \circ \Phi_z(\mathcal{H}_t(f)) && \text{(by homogeneity in } \mathbf{y}\text{)}
\end{aligned}$$

And it extends to higher degree terms just from the fact that Φ and Φ_z are homomorphisms and that Φ does not change the degree (in $\mathbf{x}$ and $\mathbf{y}$).

Further, note that if $h(a_1, \ldots, a_m) = \sum_\mathbf{e} h_\mathbf{e} \cdot \mathbf{a}^\mathbf{e}$ then

$$h' = \sum_\mathbf{e} \Phi_z(h_\mathbf{e}) \cdot \mathbf{a}^\mathbf{e}.$$

Thus,
$$\begin{aligned}
& h'(\mathcal{H}_t(g_1), \ldots, \mathcal{H}_t(g_m)) \\
&= \sum_\mathbf{e} \Phi_z(h_\mathbf{e}) \cdot (\mathcal{H}_t(\mathbf{f} \circ \Phi))^\mathbf{e} \\
&= \sum_\mathbf{e} \Phi_z(h_\mathbf{e}) \cdot \Phi \circ \Phi_z(\mathcal{H}_t(\mathbf{f})^\mathbf{e}) \\
&= \sum_\mathbf{e} (\Phi \circ \Phi_z(h_\mathbf{e})) \cdot \Phi \circ \Phi_z(\mathcal{H}_t(\mathbf{f})^\mathbf{e}) && (h_\mathbf{e} \text{ is independent of } \mathbf{y}) \\
&= \Phi \circ \Phi_z \left(\sum_\mathbf{e} h_\mathbf{e} \cdot \mathcal{H}_t(\mathbf{f})^\mathbf{e} \right) && (\Phi \text{ and } \Phi_z \text{ are homomorphisms}) \\
&= \Phi \circ \Phi_z(h(\mathcal{H}_t(f_1), \ldots, \mathcal{H}_t(f_m))) && \square
\end{aligned}$$

Corollary 7.5.3 (Matrix representation of the evolution)**.** *Suppose* A' *is a matrix whose columns are indexed by monomials in* $\mathbf{y}$*. Further suppose a row in* A' *corresponds to a polynomial, say* $h'(\mathcal{H}_t(\mathbf{g})) = h'(\mathcal{H}_t(g_1), \ldots, \mathcal{H}_t(g_m)) \in \mathbb{F}(\mathbf{g}(\mathbf{v}))[\mathbf{y}]$*, whose entry in the column indexed by* $\mathbf{y}^\mathbf{e}$ *is* $\mathsf{coeff}_{\mathbf{y}^\mathbf{e}}(h'(\mathcal{H}_t(\mathbf{g}))) \in \mathbb{F}(\mathbf{v}, \mathbf{s})$.

If A is the corresponding matrix (having entries from $\mathbb{F}(\mathbf{z})$) with columns indexed by monomials in $\mathbf{x}$ and the corresponding row being $h(\mathcal{H}_t(f_1), \ldots, \mathcal{H}_t(f_m)) \in \mathbb{F}(\mathbf{f}(\mathbf{z}))[\mathbf{x}]$ as described in Lemma 7.5.2, then

$$A' = \Phi_z(A) \times \widetilde{M_\Phi}$$

where $\widetilde{M_\Phi}(\mathbf{x}^\mathbf{e}, \mathbf{y}^\mathbf{d}) = \mathsf{coeff}_{\mathbf{y}^\mathbf{d}}(\Phi(\mathbf{x}^\mathbf{e}))$.

Proof. Suppose $h(\mathcal{H}_t(f_1), \ldots, \mathcal{H}_t(f_m)) = \sum_\mathbf{e} h_\mathbf{e}(\mathbf{z}) \cdot \mathbf{x}^\mathbf{e}$. Then,

$$\begin{aligned}
&h'(\mathcal{H}_t(g_1), \ldots, \mathcal{H}_t(g_m)) = \Phi \circ \Phi_z(h(\mathcal{H}_t(f_1), \ldots, \mathcal{H}_t(f_m))) \quad \text{(by Lemma 7.5.2)} \\
&= \sum_\mathbf{e} h_\mathbf{e}(\Phi_z(\mathbf{z})) \cdot \Phi(\mathbf{x}^\mathbf{e}) \quad = \quad \sum_\mathbf{e} h_\mathbf{e}(\Phi_z(\mathbf{z})) \cdot \left(\sum_\mathbf{d} \mathsf{coeff}_{\mathbf{y}^\mathbf{d}}(\Phi(\mathbf{x}^\mathbf{e})) \cdot \mathbf{y}^\mathbf{d} \right) \\
&= \sum_\mathbf{d} \left(\sum_\mathbf{e} h_\mathbf{e}(\Phi_z(\mathbf{z})) \cdot \mathsf{coeff}_{\mathbf{y}^\mathbf{d}}(\Phi(\mathbf{x}^\mathbf{e})) \right) \cdot \mathbf{y}^\mathbf{d}
\end{aligned}$$

Thus, the coefficient of $\mathbf{y}^\mathbf{d}$ in $h'(\mathcal{H}_t(g_1), \ldots, \mathcal{H}_t(g_m))$ is

$$\sum_\mathbf{e} \Phi_z(h_\mathbf{e}(\mathbf{z})) \cdot \mathsf{coeff}_{\mathbf{y}^\mathbf{d}}(\Phi(\mathbf{x}^\mathbf{e}))$$

which gives the required matrix decomposition. □

We are now in a position to prove Lemma 7.5.1.

Proof of Lemma 7.5.1. Without loss of generality, say $\{f_1, \ldots, f_k\}$ is an algebraically independent set. We wish to show that if $g_i = f_i \circ \Phi$, then $\{g_1, \ldots, g_k\}$ is an $\mathbb{F}$-algebraically independent set as well. Assume on the contrary that $\{g_1, \ldots, g_k\}$ is an $\mathbb{F}$-algebraically dependent set. Then for t being the inseparable degree of $\{f_1, \ldots, f_k\}$, by Lemma 6.3.8, there exists

$$h' \in \mathcal{V}_t(g_1, \ldots, g_k) := \langle \mathcal{H}_t(g_1), \ldots, \mathcal{H}_t(g_k) \rangle^{\geq 2}_{\mathbb{F}(\mathbf{g}(\mathbf{v}))} \bmod \langle \mathbf{y} \rangle^{t+1}$$

such that $\mathsf{PSSJac}_t(\mathbf{g}, h')$ is not full rank. Without loss of generality, we can assume that the entries of $\mathsf{PSSJac}_t(\mathbf{g}, h')$ are denominator-free by clearing out any denominators. Corresponding to h', define h as in Lemma 7.5.2, which would also satisfy that

$$h \in \mathcal{U}_t(f_1, \ldots, f_k) := \langle \mathcal{H}_t(f_1), \ldots, \mathcal{H}_t(f_k) \rangle^{\geq 2}_{\mathbb{F}(\mathbf{z})} \bmod \langle \mathbf{x} \rangle^{t+1}.$$

It is worth stressing the fact that the polynomial h is independent of the variable s. Then by Corollary 7.5.3 we get

$$\mathsf{PSSJac}_t(\mathbf{g}, h') = \Phi_z(\mathsf{PSSJac}_t(\mathbf{f}, h)) \times \widetilde{M_\Phi}.$$

Now, if we substitute $v_0 = 1$ and $v_i = 0$ for every $i \in [k]$, we get

$$\mathsf{PSSJac}_t(\mathbf{g}, h')(v_0 = 1, v_1 = \ldots = v_k = 0) = \mathsf{PSSJac}_t(\mathbf{f}, h)(\mathbf{z} = \mathbf{G}(\alpha)) \times \widetilde{M_\Phi}.$$

But since $\{f_1, \ldots, f_k\}$ is algebraically independent, Theorem 6.3.7 yields that $\mathsf{PSSJac}_t(\mathbf{f}, h)$ has full rank. Thus, for the correct choice of α, $\mathsf{PSSJac}_t(\mathbf{f}, h)(\mathbf{z} = \mathbf{G}(\alpha))$ also has full rank by the property we assumed $\mathcal{G}$ has.

Most crucially, the matrix $\mathsf{PSSJac}_t(\mathbf{f}, h)$ is independent of the variable s. To complete the proof, we need to show that multiplication by $\widetilde{M_\Phi}$ continues to keep this full rank to contradict the initial assumption that $\mathsf{PSSJac}_t(\mathbf{g}, h')$ was not full rank.

Finally note that for the Φ we have defined, $\widetilde{M_\Phi}$ restricted to only the *pure monomial* columns

$$\left\{ y_i^j \ : \ i \in \{1, \ldots, k\} \ , \ j \in \{0, 1, \ldots, t\} \right\},$$

is the same as M_Φ as defined in Lemma 7.4.2. Further, w is an isolating weight assignment for the set of n-variate monomials of degree at most t, we satisfy the requirements of Lemma 7.4.2. Hence, by Lemma 7.4.2,

$$\begin{aligned} &\operatorname{rank}_{\mathbb{F}(s,\alpha)} \left(\mathsf{PSSJac}_t(\mathbf{g}, h')(v_0 = 1, v_1 = \ldots, v_k = 0)\right) \\ &\qquad\qquad = \operatorname{rank}_{\mathbb{F}(\alpha)} \mathsf{PSSJac}_t(\mathbf{f}, h)(\mathbf{z} = \mathbf{G}(\alpha)) \\ \implies &\operatorname{rank}_{\mathbb{F}(s,\alpha,\mathbf{v})} \left(\mathsf{PSSJac}_t(\mathbf{g}, h')\right) \geq \operatorname{rank}_{\mathbb{F}(\alpha)} \mathsf{PSSJac}_t(\mathbf{f}, h)(\mathbf{z} = \mathbf{G}(\alpha)) = k, \end{aligned}$$

which contradicts our assumption that it was not full rank. Hence, it must indeed be the case that $\{f_1 \circ \Phi, \ldots, f_k \circ \Phi\}$ is $\mathbb{F}$ - algebraically independent. □

7.6 Explicit Faithful Homomorphisms and PIT Applications in Restricted Settings

We now describe some specific instantiations of the recipe given by Lemma 7.5.1 in restricted settings. Let us first recall the statement of the main theorem.

Theorem 7.2.1. *Let $f_1, \ldots, f_m \in \mathbb{F}[x_1, \ldots, x_n]$ be such that* $\mathsf{algrank}\{f_1, \ldots, f_m\} = k$ *and the inseparable degree is t. If t and k are bounded by a constant, then we can construct a polynomial (in the input length) sized list of homomorphisms of the form $\Phi : \mathbb{F}[\mathbf{x}] \to \mathbb{F}(s)[y_0, y_1, \ldots, y_k]$ such that at least one of them is guaranteed to be to $\mathbb{F}$-faithful for the set $\{f_1, \ldots, f_m\}$, in the following two settings:*

- *each of the f_i's are sparse polynomials,*
- *each of the f_i's are products of variable disjoint, multilinear, sparse polynomials.*

Proof. By Lemma 7.5.1, $\Phi : \mathbb{F}[x_1, \ldots, x_n] \to \mathbb{F}(s, \alpha)[y_0, \ldots, y_k]$ defined as

$$\Phi : x_i \mapsto y_0 G_i(\alpha) + \sum_{j=1}^{k} y_j \cdot s^{w(i)j},$$

is a faithful homomorphism for the set $\{f_1, \ldots, f_m\}$ if for any $h \in \mathcal{U}_t(\mathbf{f})$, $w = (w_1, \ldots, w_n)$ is a basis isolating weight assignment for $\mathsf{PSSJac}(\mathbf{f}, h)$ and the map $\mathcal{G} = (G_1(\alpha), \ldots, G_n(\alpha))$ is such that the rank of $\mathsf{PSSJac}_t(\mathbf{f}, h)$ is preserved after the substitution $\mathbf{z} \to \mathbf{a}$ for some $\mathbf{a} \in \mathcal{G}$. We define the weight using the standard hashing techniques [KS01; AB03].

Defining w: Define $w : [n] \to \mathbb{N}$ as

$$w(i) = (t+1)^i \pmod{p}$$

where t is the inseparable degree. Assuming t to be a constant, there are only $\mathrm{poly}(n)$ many distinct monomials in $\mathbf{x}$ of degree at most t. Thus, standard results by Klivans and Spielman [KS01] or Agrawal and Biswas [AB03] shows that it suffices to go over $\mathrm{poly}(n)$ many 'p's before w isolates all monomials in $\mathbf{x}$ of degree at most t.
Let $\mathsf{PSSJac}_t(\mathbf{f})$ be the matrix with columns indexed by monomials in $\mathbf{x}$ of degree at most t and rows by k-variate monomials $\mathbf{a}^{\mathbf{e}}$ in degree at most t, defined as follows.

$$\mathsf{PSSJac}_t(\mathbf{f})[\mathbf{a}^{\mathbf{e}}, \mathbf{x}^{\mathbf{d}}] = \mathsf{coeff}_{\mathbf{x}^{\mathbf{d}}}(\mathcal{H}_t(\mathbf{f})^{\mathbf{e}})$$

Set $K = \binom{k+t}{t}$ be the number of rows in $\mathsf{PSSJac}_t(\mathbf{f})$. Then the following is true.

Claim 7.6.1. *If $\mathcal{G}$ is a hitting set generator for every $K' \times K'$ minor of* $\mathsf{PSSJac}_t(\mathbf{f})$ *where $K' \leq K$, then the rank of* $\mathsf{PSSJac}_t(\mathbf{f}, h)$ *is preserved for every $h \in \mathcal{U}_t(\mathbf{f})$.*

Proof. We need to show that there is an $\mathbf{a}$ in $\mathcal{G}$ which has the following property:

For any $h \in \mathcal{U}_t(\mathbf{f})$, if $\{H_t(f_1)+h, H_t(f_2), \ldots, H_t(f_k)\}$ are linearly independent, then so are $\{H_t(f_1)(\mathbf{a})+h(\mathbf{a}), H_t(f_2)(\mathbf{a}), \ldots, H_t(f_k)(\mathbf{a})\}$.

Now suppose this is not the case. Then it must be the case that without loss of generality, some $h \in \mathcal{U}_t(\mathbf{f})$, $\mathsf{PSSJac}_t(\mathbf{f}, h)$ has full rank but for any $\mathbf{a} \in \mathcal{G}$,

$$\alpha_1(H_t(f_1)(\mathbf{a})+h(\mathbf{a})) + \sum_{i=2}^{k}(\alpha_i \cdot H_t(f_i)(\mathbf{a})) = 0.$$

Here, not all of $\{\alpha_i\}_{i\in[k]}$ are zero. However by our hypothesis, this would mean that

$$\alpha_1(H_t(f_1)+h) + \sum_{i=2}^{k}(\alpha_i \cdot H_t(f_i)) \neq 0.$$

Let $\mathcal{B}$ be a basis of the rows in $H_t(\mathbf{f})$. Then each of $\{H_t(f_1)+h, H_t(f_2), \ldots, H_t(f_k)\}$ can be written in terms of rows in $\mathcal{B}$. Thus, the above statement can be rewritten as

$$\sum_{i=1}^{K'} \beta_i \cdot b_i = \alpha_1(H_t(f_1)+h) + \sum_{i=2}^{k}(\alpha_i \cdot H_t(f_i)) \neq 0$$

where $\{\beta_i\}_{i\in[K']}$ are some scalars and $K' = |\mathcal{B}|$.

This shows that not all $\{\beta_i\}_{i=1}^{K'}$ can be zero. Now since $\mathcal{G}$ is a hitting set generator for every $K' \times K'$ minor in $\mathsf{PSSJac}_t(\mathbf{f})$, there is some $\mathbf{a} \in \mathcal{G}$ such that $\{b_i(\mathbf{a})\}_{i\in[K']}$ continue to remain linearly independent. Thus, $\sum_{i=1}^{K'} \beta_i \times b_i(\mathbf{a})! = 0$, since not all $\{\beta_i\}_{i\in[K']}$ is zero. However, this shows that

$$\alpha_1(H_t(f_1)(\mathbf{a})+h(\mathbf{a})) + \sum_{i=2}^{k}(\alpha_i \cdot H_t(f_i)(\mathbf{a})) = \sum_{i=1}^{K'} \beta_i \times b_i(\mathbf{a}) \neq 0.$$

This contradicts our assumption, and so it must be the case that for any $h \in \mathcal{U}_t(\mathbf{f})$, the rank of $\mathsf{PSSJac}_t(\mathbf{f}, h)$ is preserved. □

Now it is only a question of finding a hitting set generator of low degree, for every $K' \times K'$ minor of $\mathsf{PSSJac}_t(\mathbf{f})$ where $K' \leq K$.

Defining $\mathcal{G}$ when f_i's are sparse: When the f_i's are s-sparse, every entry of $\mathsf{PSSJac}(\mathbf{f})$ is a sum of products of at most t Hasse-derivatives of the f_i's. Further the number of such products is at most $\binom{n+t}{t}$, and hence each entry of $\mathsf{PSSJac}(\mathbf{f})$ has sparsity at most $\binom{n+t}{t} \cdot s^t$. When k, t are constants, then any $K \times K$ minor of

$\mathsf{PSSJac}(\mathbf{f})$ has sparsity $s^{O(1)}$ and hence standard hitting-set generators for sparse polynomials [KS01; AB03] would be sufficient in this setting.

Defining $\mathcal{G}$ when f_is are products of variable disjoint, multilinear, sparse polynomials: In exactly along the same lines as Agrawal *et al.* [Agr+16], we can construct hitting-set generators for minors of $\mathsf{PSSJac}(\mathbf{f})$ when each f_i is a product of variable disjoint, multilinear, sparse polynomials.

The key observation is that when $k, t = O(1)$, any $K \times K$ minor of $\mathsf{PSSJac}(\mathbf{f})$ only involves derivatives over constantly many variables, say $x_1, \ldots, x_\ell$ with $\ell \leq Kt$. Since each f_i is a product of variable disjoint sparse polynomials, each row of this submatrix can be expressed as a common factor F and a product of ℓ sparse polynomials. The reason is as follows.

> If $f = g.g'$ where g' is independent of variables in $S \subseteq \{x_1, \ldots, x_n\}$, then for any monomial $\mathbf{x}^{\mathbf{e}}$ that depends only on S we have
>
> $$\mathsf{coeff}_{\mathbf{x}^{\mathbf{e}}}(\mathcal{H}_t(f)) = \mathsf{coeff}_{\mathbf{x}^{\mathbf{e}}}(\mathcal{H}_t(g)).g'(z).$$

Hence, the determinant of this matrix is a product of sparse polynomials (each of sparsity at most $s^{Kt} = \mathrm{poly}(s)$ when $k, t = O(1)$). Standard hitting-set generators for sparse polynomials [KS01; AB03] are sufficient in this case as well. □

Applications to PIT

Using Lemma 7.1.2, two straightforward corollaries are PIT for related models.

Corollary 7.2.2. *If $\{f_1, \ldots, f_m\} \in \mathbb{F}[x_1, \ldots, x_n]$ is a set of s-sparse polynomials with algebraic rank k and inseparable degree t where $k, t = O(1)$. Then, for the class of polynomials of the form $C(f_1, \ldots, f_m)$ for any polynomial $C(z_1, \ldots, z_m) \in \mathbb{F}[\mathbf{z}]$, there is an explicit hitting set of size $(s \cdot \deg(C))^{O(1)}$.*

Proof. Without loss of generality, we may assume that $\mathbb{F}$ is algebraically closed (since nonzeroness of polynomials remain unchanged when interpreted as polynomials over an extension). Suppose $\{f_1, \ldots, f_k\}$ be a separable transcendence basis for $\{f_1, \ldots, f_m\}$ inseparable degree t.

By Theorem 7.2.1, we have a polynomial sized list of maps

$$\{\Phi_i : \mathbb{F}[\mathbf{x}] \to \mathbb{F}[s, y_0, \ldots, y_k, \alpha]\}$$

each of degree $\mathrm{poly}(n)$ such that at least one of them is $\mathbb{F}$-faithful for $\{f_1, \ldots, f_k\}$ (and hence also for $\{f_1, \ldots, f_m\}$); let Φ be such a $\mathbb{F}$-faithful homomorphism. From the construction of Theorem 7.2.1, the homomorphism Φ has degree $\mathrm{poly}(s)$. By Lemma 7.1.2, we know that $C(f_1, \ldots, f_m) = 0$ if and only if $\Phi(C(f_1, \ldots, f_m))$ is zero. Now that $\Phi(C(f_1, \ldots, f_m))$ is a polynomial in $k + 3 = O(1)$ variables, we can use the hitting set obtained from the Polynomial Identity Lemma [Ore22; DL78; Sch80; Zip79] to give hitting set of size $\mathrm{poly}(s, \deg(C))$ for $C(f_1, \ldots, f_m)$. □

Along exactly the same lines, we get the following corollary when we are working with depth-4 multilinear circuits of small algebraic rank and inseparable degree.

Corollary 7.2.3. *Let $C = \sum_{i=1}^{m} T_i$ be a depth-4 multilinear circuit of size s, where each T_i is a product of variable-disjoint, s-sparse polynomials. Suppose $\{T_1, \ldots, T_m\} \in \mathbb{F}[x_1, \ldots, x_n]$ is a set of polynomials with algebraic rank k and inseparable degree t where $k, t = O(1)$. Then, for the class of polynomials of the form $C(T_1, \ldots, T_m)$ for any polynomial $C(z_1, \ldots, z_m) \in \mathbb{F}[\mathbf{z}]$, there is an explicit hitting set of size $(s \cdot \deg(C))^{O(1)}$.*

As mentioned in the introduction, the above result is incomparable with the PIT results of Pandey *et al.* [PSS18] and Kumar and Saraf [KS17a].

We conclude with some very natural open problems in the context of this work.

- Are the homomorphisms we constructed also $\mathbb{F}(s)$-faithful homomorphisms?

  Our proof only provides a recipe towards constructing $\mathbb{F}$-faithful homomorphisms due to technical obstacles involving the criterion for algebraic independence over finite characteristic fields. The exact point where it fails is in the proof of Lemma 7.5.1. It is crucial that $h \in \mathcal{U}_t(\mathbf{f})$ is s-free for our proof to work. This is not an issue in characteristic zero fields and Agrawal *et al.* [Agr+16] construct $\mathbb{F}(s)$-faithful homomorphisms.

- How crucial is the notion of inseparable degree in the context of testing algebraic independence?

  The criterion of Pandey, Saxena and Sinhababu [PSS18] crucially depends on this field theoretic notion and there seems to be compelling algebraic reasons to believe that this is necessary. However, as mentioned earlier, Guo, Saxena and Sinhababu [GSS19] showed that algebraic independence testing is in $\mathsf{AM} \cap \mathsf{co} - \mathsf{AM}$ and this proof has absolutely no dependence on the inseparable degree of the polynomials.

8 Conclusion

In this thesis, we looked at some central questions in the field of Algebraic Complexity Theory. The first two chapters were dedicated to proving lower bounds against general algebraic models of computation. In chapter 2, our main result was a tight $\Omega(nd)$ lower bound against algebraic branching programs for $\sum_{i=1}^{n} x_i^d$ and in chapter 3, our main result was a quadratic lower bound against algebraic formulas for the multilinear polynomial $\mathrm{ESYM}_{n,0.1n}(\mathbf{x})$. Improving these lower bounds, even in restricted settings, would be interesting. In particular proving super-linear lower bounds against ABPs computing constant degree polynomials or proving super-quadratic lower bounds against multilinear ABPs seem approachable, given our current knowledge. Similarly proving super-quadratic lower bounds against homogeneous formulas would be extremely interesting.

For the next two chapters, we were interested in the non-commutative setting. In chapter 4 we showed that weak (super-quartic) lower bounds against multilinear circuits would imply strong lower bounds against non-commutative circuits, thus providing a possible *reason* for knowing only almost quadratic lower bounds against multilinear circuits in spite of knowing super-polynomial lower bounds against multilinear formulas. An interesting question is whether super-quadratic lower bounds against multilinear circuits imply strong lower bounds against non-commutative circuits. The motivating question for chapter 5 was Nisan's conjecture [Nis91] that there is a super-polynomial separation between the powers of formulas and ABPs in the non-commutative setting. We made progress towards this by proving a tight super-polynomial separation between the ABPs and some structured formulas in this setting. Resolving Nisan's conjecture is the most interesting related question.

In the second part, we gave better upper bounds for the question of polynomial identity testing (PIT) in certain settings via improved algebraic independence testing algorithms [PSS18] over fields of finite characteristic. In chapter 6, we studied the question of testing algebraic independence of a given set of polynomials and then in chapter 7, we constructed *faithful homomorphisms* in certain settings. As a corollary, this allowed us to give efficient PIT algorithms in related settings. Improving the upper bound on algebraic independence testing over finite fields in more general settings is an interesting open problem.

Bibliography

[AB03] Manindra Agrawal and Somenath Biswas. “Primality and identity testing via Chinese remaindering”. In: *J. ACM* 50.4 (2003), pp. 429–443. URL: http://doi.acm.org/10.1145/792538.792540 (cit. on pp. 123, 131, 133).

[Agr+16] Manindra Agrawal, Chandan Saha, Ramprasad Saptharishi, and Nitin Saxena. “Jacobian Hits Circuits: Hitting Sets, Lower Bounds for Depth-D Occur-k Formulas and Depth-3 Transcendence Degree-k Circuits”. In: *SIAM J. Comput.* 45.4 (2016), pp. 1533–1562. URL: https://doi.org/10.1137/130910725 (cit. on pp. 18, 19, 118–120, 133, 134).

[AV08] Manindra Agrawal and V Vinay. “Arithmetic circuits: A chasm at depth four”. In: *2008 49th Annual IEEE Symposium on Foundations of Computer Science*. IEEE. 2008, pp. 67–75 (cit. on p. 15).

[AKV20] Noga Alon, Mrinal Kumar, and Ben Lee Volk. “Unbalancing Sets and An Almost Quadratic Lower Bound for Syntactically Multilinear Arithmetic Circuits”. In: *Comb.* 40.2 (2020), pp. 149–178. URL: https://doi.org/10.1007/s00493-019-4009-0 (cit. on pp. 12, 76).

[AB09] Sanjeev Arora and Boaz Barak. *Computational complexity: a modern approach*. Cambridge University Press, 2009 (cit. on p. 113).

[AJR16] V. Arvind, P. S. Joglekar, and S. Raja. “Noncommutative Valiant’s Classes: Structure and Complete Problems”. In: *ACM Trans. Comput. Theory* 9.1 (2016). URL: https://doi.org/10.1145/2956230 (cit. on p. 67).

[AS18] Vikraman Arvind and Srikanth Srinivasan. “On the hardness of the noncommutative determinant”. In: *Comput. Complex.* 27.1 (2018), pp. 1–29. URL: https://doi.org/10.1007/s00037-016-0148-5 (cit. on p. 68).

[BS83] Walter Baur and Volker Strassen. “The Complexity of Partial Derivatives”. In: *Theor. Comput. Sci.* 22 (1983), pp. 317–330. URL: https://doi.org/10.1016/0304-3975(83)90110-X (cit. on pp. 10, 11, 13, 25, 27, 53, 77).

[BMS13] Malte Beecken, Johannes Mittmann, and Nitin Saxena. “Algebraic independence and blackbox identity testing”. In: *Inf. Comput.* 222 (2013), pp. 2–19. URL: https://doi.org/10.1016/j.ic.2012.10.004 (cit. on pp. 18, 19, 117–119).

[Ben83] Michael Ben-Or. “Lower Bounds for Algebraic Computation Trees (Preliminary Report)”. In: *Proceedings of the 15th Annual ACM Symposium on Theory of Computing, 25-27 April, 1983, Boston, Massachusetts, USA*. ACM, 1983, pp. 80–86. URL: https://doi.org/10.1145/800061.808735 (cit. on pp. 11, 27).

[Ben94] Michael Ben-Or. “Algebraic Computation Trees in Characteristi p>0 (Extended Abstract)”. In: *35th Annual Symposium on Foundations of Computer Science, Santa Fe, New Mexico, USA, 20-22 November 1994*. IEEE Computer Society, 1994, pp. 534–539. URL: https://doi.org/10.1109/SFCS.1994.365738 (cit. on p. 27).

[BC92] Michael Ben-Or and Richard Cleve. “Computing Algebraic Formulas Using a Constant Number of Registers”. In: *SIAM J. Comput.* 21.1 (1992), pp. 54–58. URL: https://doi.org/10.1137/0221006 (cit. on p. 96).

[Bre74] Richard P. Brent. “The Parallel Evaluation of General Arithmetic Expressions”. In: *J. ACM* 21.2 (1974), pp. 201–206 (cit. on pp. 14, 87, 88).

[Car+18] Marco L. Carmosino, Russell Impagliazzo, Shachar Lovett, and Ivan Mihajlin. “Hardness Amplification for Non-Commutative Arithmetic Circuits”. In: *33rd Computational Complexity Conference, CCC*. Ed. by Rocco A. Servedio. Vol. 102. LIPIcs. 2018, 12:1–12:16. URL: https://doi.org/10.4230/LIPIcs.CCC.2018.12 (cit. on pp. 67, 74).

[Cha21] Prerona Chatterjee. “Separating ABPs and Some Structured Formulas in the Non-Commutative Setting”. In: *36th Computational Complexity Conference, CCC 2021*. Vol. 200. LIPIcs. 2021, 7:1–7:24. URL: https://doi.org/10.4230/LIPIcs.CCC.2021.7 (cit. on pp. 1, 16).

[Cha+20] Prerona Chatterjee, Mrinal Kumar, Adrian She, and Ben Lee Volk. “A Quadratic Lower Bound for Algebraic Branching Programs”. In: *35th Computational Complexity Conference, CCC 2020, July 28-31, 2020, Saarbrücken, Germany (Virtual Conference)*. Ed. by Shubhangi Saraf. Vol. 169. LIPIcs. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2020, 2:1–2:21. URL: https://doi.org/10.4230/LIPIcs.CCC.2020.2 (cit. on pp. 1, 16).

[CS19] Prerona Chatterjee and Ramprasad Saptharishi. “Constructing Faithful Homomorphisms over Fields of Finite Characteristic”. In: *39th IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science*. Vol. 150. LIPIcs. 2019, 11:1–11:14. URL: https://doi.org/10.4230/LIPIcs.FSTTCS.2019.11 (cit. on pp. 1, 20).

[CM17] Suryajith Chillara and Partha Mukhopadhyay. “On the limits of depth reduction at depth 3 over small finite fields”. In: *Information and Computation* 256 (2017), pp. 35–44 (cit. on p. 15).

[CLO07] David Cox, John Little, and Donal O’Shea. *Ideals, Varieties and Algorithms*. Undergraduate texts in mathematics. Springer, 2007. URL: https://link.springer.com/book/10.1007/978-3-319-16721-3 (cit. on pp. 30, 57).

[DL78] Richard A. DeMillo and Richard J. Lipton. “A Probabilistic Remark on Algebraic Program Testing”. In: *Inf. Process. Lett.* 7.4 (1978), pp. 193–195. URL: https://doi.org/10.1016/0020-0190(78)90067-4 (cit. on pp. 17, 106, 134).

[DGW09] Zeev Dvir, Ariel Gabizon, and Avi Wigderson. “Extractors And Rank Extractors For Polynomial Sources”. In: *Comput. Complex.* 18.1 (2009), pp. 1–58. URL: https://doi.org/10.1007/s00037-009-0258-4 (cit. on p. 112).

[Dvi+12] Zeev Dvir, Guillaume Malod, Sylvain Perifel, and Amir Yehudayoff. “Separating multilinear branching programs and formulas”. In: *Proceedings of the 44th Symposium on Theory of Computing Conference, STOC 2012, New York, NY, USA, May 19 - 22, 2012*. Ed. by Howard J. Karloff and Toniann Pitassi. ACM, 2012, pp. 615–624. URL: https://doi.org/10.1145/2213977.2214034 (cit. on pp. 12, 23, 70).

[EGS75] Paul Erdős, Ronald L. Graham, and Endre Szemerédi. “On sparse graphs with dense long paths”. In: *Computers & Mathematics with Applications* 1.3 (1975), pp. 365 –369. URL: http://www.sciencedirect.com/science/article/pii/0898122175900371 (cit. on p. 45).

[Fij+20] Nathanaël Fijalkow, Guillaume Lagarde, Pierre Ohlmann, and Olivier Serre. “Lower Bounds for Arithmetic Circuits via the Hankel Matrix”. In: *37th International Symposium on Theoretical Aspects of Computer Science, STACS*. Vol. 154. LIPIcs. 2020, 24:1–24:17. URL: https://doi.org/10.4230/LIPIcs.STACS.2020.24 (cit. on p. 67).

[Fou+15] Hervé Fournier, Nutan Limaye, Guillaume Malod, and Srikanth Srinivasan. “Lower bounds for depth-4 formulas computing iterated matrix multiplication”. In: *SIAM Journal on Computing* 44.5 (2015), pp. 1173–1201 (cit. on p. 15).

[GR08] Ariel Gabizon and Ran Raz. “Deterministic extractors for affine sources over large fields”. In: *Comb.* 28.4 (2008), pp. 415–440. URL: https://doi.org/10.1007/s00493-008-2259-3 (cit. on pp. 121, 123).

[GK98] Dima Grigoriev and Marek Karpinski. “Computing the Additive Complexity of Algebraic Circuits with Root Extracting”. In: *SIAM J. Comput.* 27.3 (1998), pp. 694–701. URL: https://doi.org/10.1137/S0097539793258313 (cit. on p. 15).

[GR00] Dima Grigoriev and Alexander Razborov. “Exponential lower bounds for depth 3 arithmetic circuits in algebras of functions over finite fields”. In: *Applicable Algebra in Engineering, Communication and Computing* 10.6 (2000), pp. 465–487 (cit. on p. 15).

[GSS19] Zeyu Guo, Nitin Saxena, and Amit Sinhababu. “Algebraic Dependencies and PSPACE Algorithms in Approximative Complexity over Any Field”. In: *Theory Comput.* 15 (2019), pp. 1–30. URL: https://doi.org/10.4086/toc.2019.v015a016 (cit. on pp. 16, 106, 112, 113, 134).

[Gup+14] Ankit Gupta, Pritish Kamath, Neeraj Kayal, and Ramprasad Saptharishi. “Approaching the chasm at depth four”. In: *Journal of the ACM (JACM)* 61.6 (2014), pp. 1–16. URL: http://dx.doi.org/10.1145/2629541 (cit. on p. 15).

[Gup+16] Ankit Gupta, Pritish Kamath, Neeraj Kayal, and Ramprasad Saptharishi. “Arithmetic Circuits: A Chasm at Depth 3”. In: *SIAM J. Comput.* 45.3 (2016), pp. 1064–1079. URL: https://doi.org/10.1137/140957123 (cit. on pp. 15, 26).

[GST20] Nikhil Gupta, Chandan Saha, and Bhargav Thankey. “A Super-Quadratic Lower Bound for Depth Four Arithmetic Circuits”. In: *35th Computational Complexity Conference, CCC 2020, July 28-31, 2020, Saarbrücken, Germany (Virtual Conference)*. Ed. by Shubhangi Saraf. Vol. 169. LIPIcs. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2020, 23:1–23:31. URL: https://doi.org/10.4230/LIPIcs.CCC.2020.23 (cit. on p. 15).

[HW15] Pavel Hrubes and Avi Wigderson. “Non-Commutative Arithmetic Circuits with Division”. In: *Theory Comput.* 11 (2015), pp. 357–393. URL: https://doi.org/10.4086/toc.2015.v011a014 (cit. on pp. 84, 87, 88).

[HWY10] Pavel Hrubes, Avi Wigderson, and Amir Yehudayoff. “Relationless Completeness and Separations”. In: *Proceedings of the 25th Annual IEEE Conference on Computational Complexity, CCC*. IEEE Computer Society, 2010, pp. 280–290. URL: https://doi.org/10.1109/CCC.2010.34 (cit. on p. 67).

[HWY11] Pavel Hrubeš, Avi Wigderson, and Amir Yehudayoff. “Non-commutative circuits and the sum-of-squares problem”. In: *Journal of the American Mathematical Society* 24.3 (2011), pp. 871–898. URL: https://www.ams.org/journals/jams/2011-24-03/S0894-0347-2011-00694-2/S0894-0347-2011-00694-2.pdf (cit. on pp. 67, 68, 71, 74).

[HY11] Pavel Hrubes and Amir Yehudayoff. “Homogeneous Formulas and Symmetric Polynomials”. In: *Comput. Complex.* 20.3 (2011), pp. 559–578. URL: https://doi.org/10.1007/s00037-011-0007-3 (cit. on pp. 85, 86, 99).

[HY16] Pavel Hrubes and Amir Yehudayoff. “On Isoperimetric Profiles and Computational Complexity”. In: *43rd International Colloquium on Automata, Languages, and Programming, ICALP 2016, July 11-15, 2016, Rome, Italy*. Ed. by Ioannis Chatzigiannakis, Michael Mitzenmacher, Yuval Rabani, and Davide Sangiorgi. Vol. 55. LIPIcs. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2016, 89:1–89:12. URL: https://doi.org/10.4230/LIPIcs.ICALP.2016.89 (cit. on p. 23).

[Hya77] L. Hyafil. “The power of commutativity”. In: *18th Annual Symposium on Foundations of Computer Science (sfcs 1977)*. 1977, pp. 171–174 (cit. on p. 67).

[Hya79] Laurent Hyafil. “On the parallel evaluation of multivariate polynomials”. In: *SIAM Journal on Computing* 8.2 (1979), pp. 120–123. URL: http://dx.doi.org/10.1137/0208010 (cit. on p. 14).

[Isa94] Irving Martin Isaacs. *Character theory of finite groups*. New York: Dover publications Inc., 1994 (cit. on p. 107).

[Jac41] C.G.J. Jacobi. “De Determinantibus functionalibus.” lat. In: *Journal für die reine und angewandte Mathematik* 22 (1841), pp. 319–359. URL: http://eudml.org/doc/147138 (cit. on pp. 16, 106).

[Juk12] Stasys Jukna. *Boolean Function Complexity - Advances and Frontiers*. Vol. 27. Algorithms and combinatorics. Springer, 2012. URL: https://doi.org/10.1007/978-3-642-24508-4 (cit. on p. 52).

[Kal85] Kyriakos Kalorkoti. “A Lower Bound for the Formula Size of Rational Functions”. In: *SIAM Journal on Computing* 14.3 (1985), pp. 678–687 (cit. on pp. 11, 52, 53).

[KW93] Mauricio Karchmer and Avi Wigderson. “On Span Programs”. In: *Proceedings of the Eigth Annual Structure in Complexity Theory Conference, San Diego, CA, USA, May 18-21, 1993*. IEEE Computer Society, 1993, pp. 102–111. URL: https://doi.org/10.1109/SCT.1993.336536 (cit. on p. 26).

[Kay09] Neeraj Kayal. “The Complexity of the Annihilating Polynomial”. In: *Proceedings of the 24th Annual IEEE Conference on Computational Complexity, CCC 2009, Paris, France, 15-18 July 2009*. 2009, pp. 184–193. URL: https://doi.org/10.1109/CCC.2009.37 (cit. on pp. 16, 105).

[Kay+17] Neeraj Kayal, Nutan Limaye, Chandan Saha, and Srikanth Srinivasan. “An exponential lower bound for homogeneous depth four arithmetic formulas”. In: *SIAM Journal on Computing* 46.1 (2017), pp. 307–335 (cit. on p. 15).

[KS16] Neeraj Kayal and Chandan Saha. “Lower bounds for depth-three arithmetic circuits with small bottom fanin”. In: *computational complexity* 25.2 (2016), pp. 419–454 (cit. on p. 15).

[KS01] Adam R. Klivans and Daniel A. Spielman. “Randomness efficient identity testing of multivariate polynomials”. In: *Proceedings on 33rd Annual ACM Symposium on Theory of Computing, July 6-8, 2001, Heraklion, Crete, Greece*. Ed. by Jeffrey Scott Vitter, Paul G. Spirakis, and Mihalis Yannakakis. ACM, 2001, pp. 216–223. URL: https://doi.org/10.1145/380752.380801 (cit. on pp. 123, 131, 133).

[Kna07] Anthony W Knapp. *Advanced algebra*. Springer Science & Business Media, 2007 (cit. on p. 110).

[Koi12] Pascal Koiran. “Arithmetic circuits: The chasm at depth four gets wider”. In: *Theoretical Computer Science* 448 (2012), pp. 56–65. URL: http://dx.doi.org/10.1016/j.tcs.2012.03.041 (cit. on p. 15).

[Kum19] Mrinal Kumar. “A quadratic lower bound for homogeneous algebraic branching programs”. In: *Computational Complexity* 28.3 (2019), pp. 409–435. URL: https://doi.org/10.1007/s00037-019-00186-3 (cit. on pp. 26, 27, 31, 33, 34).

[KS19] Mrinal Kumar and Ramprasad Saptharishi. “The Computational Power of Depth Five Arithmetic Circuits”. In: *SIAM J. Comput.* 48.1 (2019), pp. 144–180. URL: https://doi.org/10.1137/18M1173319 (cit. on p. 15).

[KS15] Mrinal Kumar and Shubhangi Saraf. “The limits of depth reduction for arithmetic formulas: It’s all about the top fan-in”. In: *SIAM Journal on Computing* 44.6 (2015), pp. 1601–1625 (cit. on p. 15).

[KS17a] Mrinal Kumar and Shubhangi Saraf. “Arithmetic Circuits with Locally Low Algebraic Rank”. In: *Theory of Computing* 13.1 (2017), pp. 1–33. URL: https://doi.org/10.4086/toc.2017.v013a006 (cit. on pp. 120, 134).

[KS17b] Mrinal Kumar and Shubhangi Saraf. “On the Power of Homogeneous Depth 4 Arithmetic Circuits”. In: *SIAM J. Comput.* 46.1 (2017), pp. 336–387. URL: https://doi.org/10.1137/140999335 (cit. on p. 15).

[LLS19] Guillaume Lagarde, Nutan Limaye, and Srikanth Srinivasan. “Lower Bounds and PIT for Non-commutative Arithmetic Circuits with Restricted Parse Trees”. In: *Comput. Complex.* 28.3 (2019), pp. 471–542. URL: https://doi.org/10.1007/s00037-018-0171-9 (cit. on pp. 67, 73, 76, 77, 90).

[LMP19] Guillaume Lagarde, Guillaume Malod, and Sylvain Perifel. “Non-commutative computations: lower bounds and polynomial identity testing”. In: *Chic. J. Theor. Comput. Sci.* 2019 (2019). URL: http://cjtcs.cs.uchicago.edu/articles/2019/2/contents.html (cit. on pp. 54, 56, 67, 70).

[LMS16] Nutan Limaye, Guillaume Malod, and Srikanth Srinivasan. “Lower Bounds for Non-Commutative Skew Circuits”. In: *Theory of Computing* 12.1 (2016), pp. 1–38. URL: https://doi.org/10.4086/toc.2016.v012a012 (cit. on p. 67).

[LST21a] Nutan Limaye, Srikanth Srinivasan, and Sébastien Tavenas. “New Non-FPT Lower Bounds for Some Arithmetic Formulas”. In: *Electron. Colloquium Comput. Complex.* 28 (2021), p. 94. URL: https://eccc.weizmann.ac.il/report/2021/094 (cit. on pp. 14, 67, 77, 84, 100).

[LST21b] Nutan Limaye, Srikanth Srinivasan, and Sébastien Tavenas. “Superpolynomial Lower Bounds Against Low-Depth Algebraic Circuits”. In: *Electron. Colloquium Comput. Complex.* 28 (2021), p. 81. URL: https://eccc.weizmann.ac.il/report/2021/081 (cit. on p. 15).

[MV97] Meena Mahajan and V. Vinay. “Determinant: Combinatorics, Algorithms, and Complexity”. In: *Chicago J. Theor. Comput. Sci.* 1997 (1997). URL: http://cjtcs.cs.uchicago.edu/articles/1997/5/contents.html (cit. on p. 70).

[MZ17] Izaak Meckler and Gjergji Zaimi. *Singular locus of zero locus of elementary symmetric polynomials*. 2017. URL: https://mathoverflow.net/questions/264226/singular-locus-of-zero-set-of-elementary-symmetric-polynomial (cit. on pp. 54, 56).

[MW19] Merriam and Webster. *Definition of Abecedarian*. Word of the Day at www.merriam-webster.com. 2019. URL: https://www.merriam-webster.com/word-of-the-day/abecedarian-2019-03-06 (cit. on p. 68).

[MSS14] Johannes Mittmann, Nitin Saxena, and Peter Scheiblechner. “Algebraic independence in positive characteristic: A p-adic calculus”. In: *Transactions of the American Mathematical Society* 366.7 (2014), pp. 3425–3450 (cit. on p. 112).

[Nec66] Eduard Ivanovich Nechiporuk. “On a Boolean function”. In: *Dokl. Akad. Nauk SSSR* 169 (4 1966), pp. 765–766. URL: http://mi.mathnet.ru/dan32449 (cit. on pp. 11, 26, 52).

[Nis91] Noam Nisan. “Lower Bounds for Non-Commutative Computation (Extended Abstract)”. In: *Proceedings of the 23rd Annual ACM Symposium on Theory of Computing*. ACM, 1991, pp. 410–418. URL: https://doi.org/10.1145/103418.103462 (cit. on pp. 13, 23, 24, 67, 77, 135).

[NW97] Noam Nisan and Avi Wigderson. “Lower Bounds on Arithmetic Circuits Via Partial Derivatives”. In: *Comput. Complex.* 6.3 (1997), pp. 217–234. URL: https://doi.org/10.1007/BF01294256 (cit. on pp. 15, 26).

[Ore22] Øystein Ore. “Über höhere Kongruenzen”. In: *Norsk Mat. Forenings Skrifter* 1.7 (1922), p. 15 (cit. on pp. 17, 106, 134).

[Oxl92] James G. Oxley. *Matroid theory*. Oxford University Press, 1992 (cit. on pp. 16, 105).

[PSS18] Anurag Pandey, Nitin Saxena, and Amit Sinhababu. “Algebraic independence over positive characteristic: New criterion and applications to locally low-algebraic-rank circuits”. In: *Comput. Complex.* 27.4 (2018), pp. 617–670. URL: https://doi.org/10.1007/s00037-018-0167-5 (cit. on pp. 17, 19, 107–110, 113, 118, 120, 121, 134, 135).

[Per27] O Perron. “Algebra I (Die Grundlagen) Göschens Lehrbücherei”. In: *Berlin und Leipzig* (1927) (cit. on p. 112).

[Raz06] Ran Raz. “Separation of Multilinear Circuit and Formula Size”. In: *Theory of Computing* 2.6 (2006), pp. 121–135. URL: https://doi.org/10.4086/toc.2006.v002a006 (cit. on pp. 12, 23).

[Raz08] Ran Raz. “Elusive functions and lower bounds for arithmetic circuits”. In: *Proceedings of the 40th Annual ACM Symposium on Theory of Computing, Victoria, British Columbia, Canada, May 17-20, 2008*. Ed. by Cynthia Dwork. ACM, 2008, pp. 711–720. URL: https://doi.org/10.1145/1374376.1374479 (cit. on p. 15).

[Raz09] Ran Raz. “Multi-linear formulas for permanent and determinant are of super-polynomial size”. In: *J. ACM* 56.2 (2009), 8:1–8:17. URL: https://doi.org/10.1145/1502793.1502797 (cit. on pp. 9, 12).

[Raz13] Ran Raz. “Tensor-Rank and Lower Bounds for Arithmetic Formulas”. In: *J. ACM* 60.6 (2013), 40:1–40:15. URL: https://doi.org/10.1145/2535928 (cit. on pp. 84, 86, 91).

[RSY08] Ran Raz, Amir Shpilka, and Amir Yehudayoff. “A Lower Bound for the Size of Syntactically Multilinear Arithmetic Circuits”. In: *SIAM J. Comput.* 38.4 (2008), pp. 1624–1647. URL: https://doi.org/10.1137/070707932 (cit. on p. 12).

[RY08] Ran Raz and Amir Yehudayoff. “Balancing Syntactically Multilinear Arithmetic Circuits”. In: *Computational Complexity* 17.4 (2008), pp. 515–535 (cit. on p. 23).

[Sap15] Ramprasad Saptharishi. “A survey of lower bounds in arithmetic circuit complexity”. Github survey. 2015. URL: https://github.com/dasarpmar/lowerbounds-survey/releases/ (cit. on p. 12).

[ST18] Ramprasad Saptharishi and Anamay Tengse. “Quasipolynomial Hitting Sets for Circuits with Restricted Parse Trees”. In: *38th IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science, FSTTCS*. Vol. 122. LIPIcs. 2018, 6:1–6:19. URL: https://doi.org/10.4230/LIPIcs.FSTTCS.2018.6 (cit. on p. 67).

[Sch80] Jacob T. Schwartz. “Fast Probabilistic Algorithms for Verification of Polynomial Identities”. In: *J. ACM* 27.4 (1980), pp. 701–717 (cit. on pp. 17, 106, 134).

[SW01] Amir Shpilka and Avi Wigderson. “Depth-3 arithmetic circuits over fields of characteristic zero”. In: *Comput. Complex.* 10.1 (2001), pp. 1–27. URL: https://doi.org/10.1007/PL00001609 (cit. on pp. 15, 52).

[SY10] Amir Shpilka and Amir Yehudayoff. “Arithmetic Circuits: A survey of recent results and open questions”. In: *Found. Trends Theor. Comput. Sci.* 5.3-4 (2010), pp. 207–388. URL: https://doi.org/10.1561/0400000039 (cit. on pp. 10, 12, 52, 53).

[Smi14] Justin R. Smith. *Introduction to Algebraic Geometry*. Textbooks in Mathematics. Taylor & Francis, 2014. URL: https://books.google.com/books?id=zx7usgEACAAJ (cit. on pp. 30, 62).

[Smo97] Roman Smolensky. “Easy Lower Bound for a Strange Computational Model”. In: *Computational Complexity* 6.3 (1997), pp. 213–216. URL: https://doi.org/10.1007/BF01294255 (cit. on p. 11).

[Spi71] Philip Spira. “On time-hardware complexity tradeoffs for Boolean functions”. In: *Proceedings of the 4th Hawaii Symposium on System Sciences, 1971*. 1971, pp. 525–527 (cit. on p. 14).

[Str73a] Volker Strassen. “Vermeidung von Divisionen.” ger. In: *Journal für die reine und angewandte Mathematik* 264 (1973), pp. 184–202. URL: http://eudml.org/doc/151394 (cit. on pp. 10, 13, 25, 27, 77).

[Str73b] Volker Strassen. “Vermeidung von Divisionen.” In: *Journal für die reine und angewandte Mathematik* 264 (1973), pp. 184–202. URL: http://eudml.org/doc/151394 (cit. on p. 26).

[Tav15] Sébastien Tavenas. “Improved bounds for reduction to depth 4 and depth 3”. In: *Information and Computation* 240 (2015), pp. 2–11 (cit. on p. 15).

[Val77] Leslie G. Valiant. “Graph-Theoretic Arguments in Low-Level Complexity”. In: *Mathematical Foundations of Computer Science 1977, 6th Symposium, Tatranska Lomnica, Czechoslovakia, September 5-9, 1977, Proceedings*. Ed. by Jozef Gruska. Vol. 53. Lecture Notes in Computer Science. Springer, 1977, pp. 162–176. URL: https://doi.org/10.1007/3-540-08353-7_135 (cit. on pp. 28, 45).

[Val79] Leslie G. Valiant. “Completeness Classes in Algebra”. In: *Proceedings of the 11h Annual ACM Symposium on Theory of Computing*. ACM, 1979, pp. 249–261. URL: https://doi.org/10.1145/800135.804419 (cit. on pp. 9, 10).

[Val+83] Leslie G. Valiant, Sven Skyum, S. Berkowitz, and Charles Rackoff. “Fast Parallel Computation of Polynomials Using Few Processors”. In: *SIAM J. Comput.* 12.4 (1983), pp. 641–644. URL: https://doi.org/10.1137/0212043 (cit. on p. 15).

[Zip79] Richard Zippel. “Probabilistic algorithms for sparse polynomials”. In: *Symbolic and Algebraic Computation, EUROSAM ’79, An International Symposiumon Symbolic and Algebraic Computation*. Vol. 72. Lecture Notes in Computer Science. Springer, 1979, pp. 216–226. URL: http://dx.doi.org/10.1007/3-540-09519-5_73 (cit. on pp. 17, 106, 134).

This thesis was typeset with LaTeX 2$_\varepsilon$. It uses the *Clean Thesis* style developed by Ricardo Langner. The design of the *Clean Thesis* style is inspired by user guide documents from Apple Inc.

Download the *Clean Thesis* style at `http://cleanthesis.der-ric.de/`.